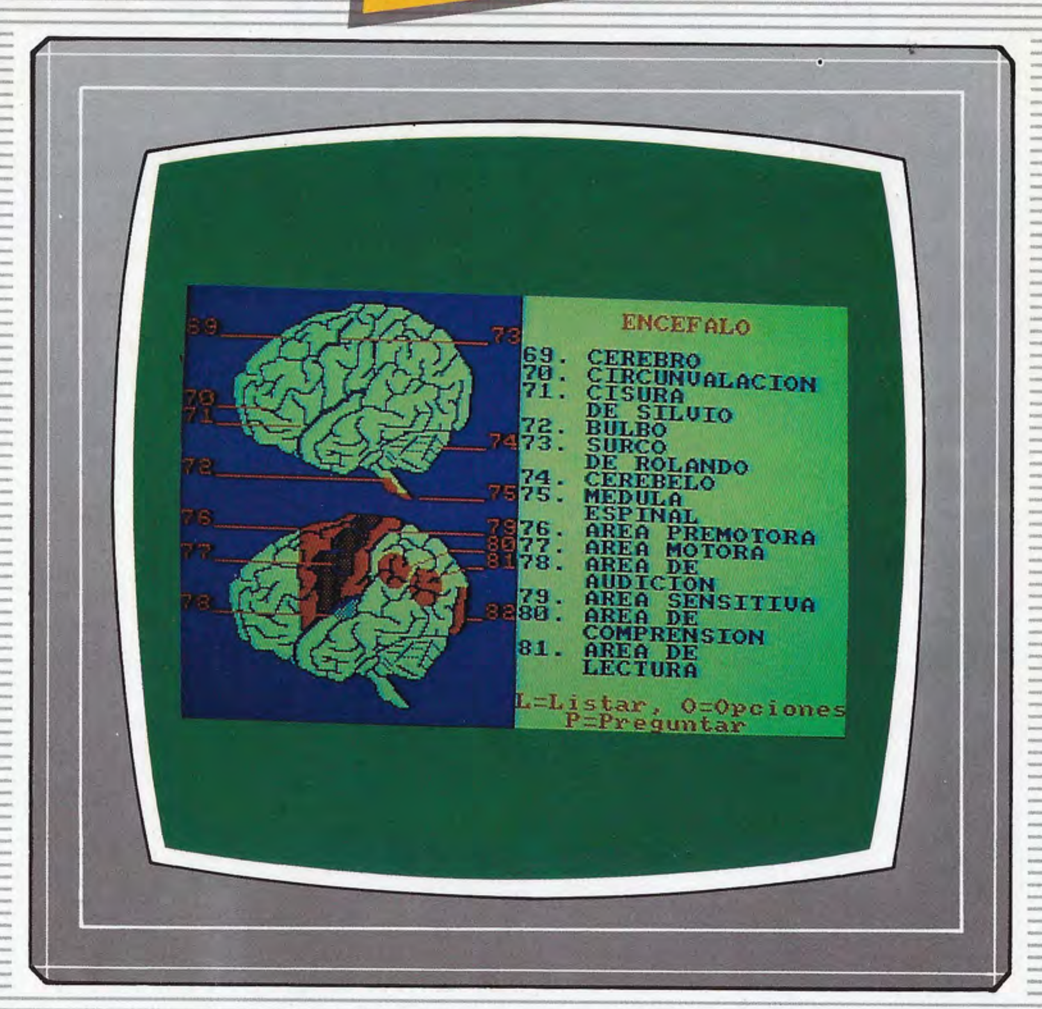
▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 8

PASO A PASO



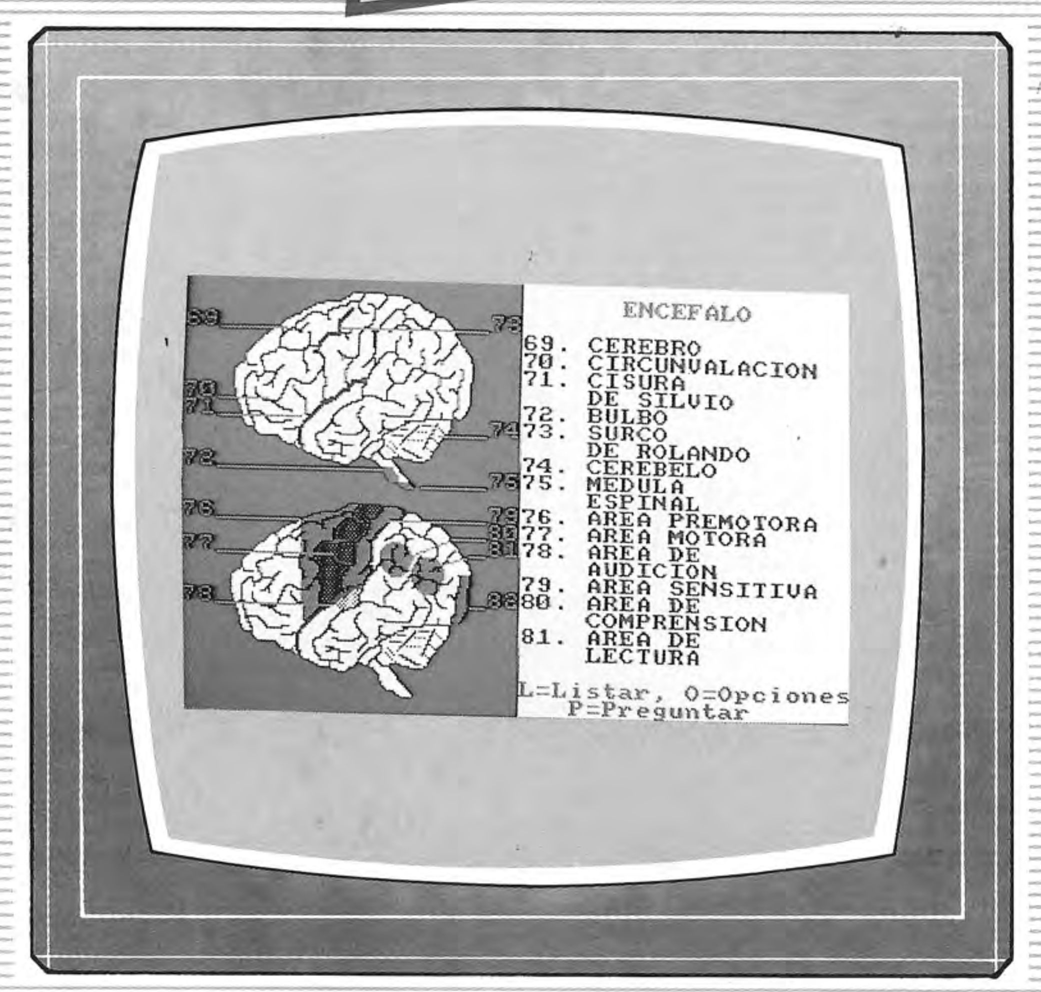
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 8

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de AULA DE INFORMATICA APLICADA (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomuniación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: Jesús Bocho, Licenciado en Informática. LOGO: Cristina Manzanero, Licenciada en Informática. APLICACIONES: Fernando Suero, Diplomado en Telecomunicación. OTROS LENGUAJES (Sistemas operativos): Domingo Villaseñor, Diplomado en Informática, y Lenguaje C: Enrique Serrano, Ingeniero en Telecomunicación.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-084-7
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

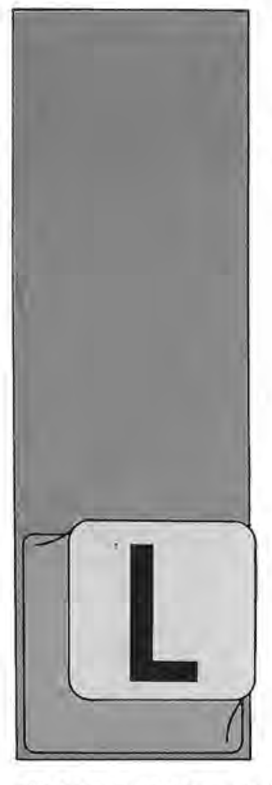
Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.
Mayo, 1987.
P.V.P. Canarias: 335,–.

INDICE

INFORMATICA BASICA

REDES DE ORDENADORES

Principios generales



LAS comunicaciones de datos se pueden establecer a través de muchas vías de comunicación o medios de transmisión. La técnica básica consiste en conectar directamente el transmisor y el receptor mediante una línea de comunicación. Como ejemplo podemos citar la conexión entre un ordenador y un terminal. Cuando aparecieron los primeros ordenadores, ésta era la única configuración posible. Pero al aumentar el número de prestaciones del ordenador, se hizo posible conectar más de un terminal.

Coincidiendo con el gran auge que tuvo el ordenador en la década de los sesenta, se desarrollaron técnicas para conectar varios ordenadores entre sí y a dispositivos que actúan como enlaces de terminales. La interconexión de varios ordenadores y terminales mediante líneas de comunicación recibe el nombre de *red de ordenadores.*

La aparición y auge de los miniordenadores en la década de los sesenta y los setenta contribuyó notablemente al desarrollo de las redes de ordenadores. Los minis eran económicamente versátiles; por consiguiente, podían utilizarse como componentes especializados de las redes. Otros tipos de redes son la red telefónica y la red postal.

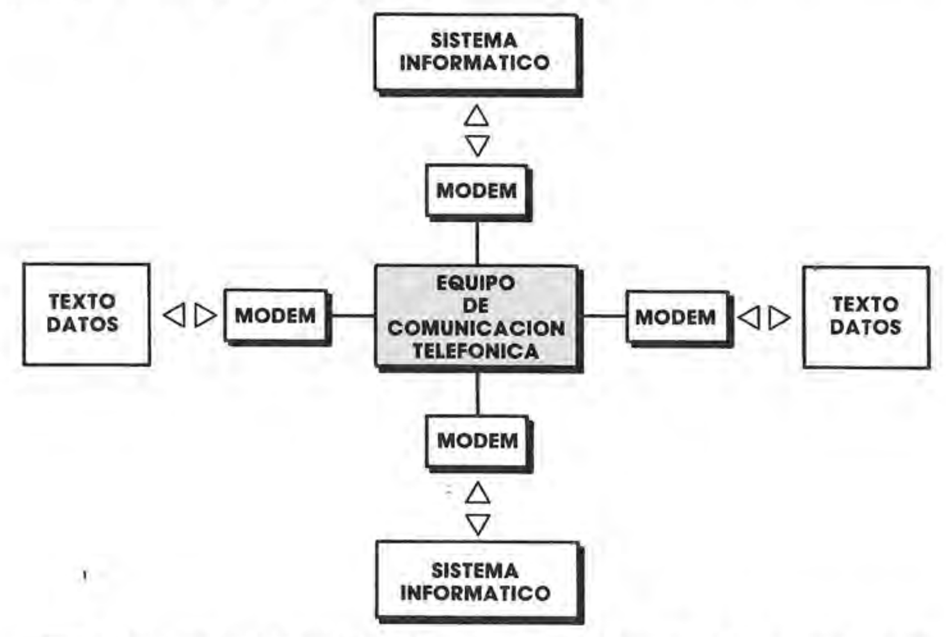
La finalidad de cualquier red de comunicaciones es transferir información entre dos puntos. Además de la información es posible, también, compartir una serie de recursos técnicos, como la CPU, unidades de almacenamiento masivo, impresoras y trazadores gráficos. Por ejemplo, una aplicación de predicción meteorológica puede ser trazada por un solo procesador o por varios procesadores ubicados en diferentes zonas geográficas.

Definiciones básicas

Una RED DE ORDENADORES se puede definir como: conjunto de ordenadores interconectados, normalmente distantes unos de otros, con el fin de repartir ciertos tipos de recursos como programas ficheros y potencia de proceso. Al principio se utilizaba la red telefónica para interconectar los ordenadores, y de hecho, todavía se utilizan para interconectar pequeños terminales u ordenadores personales; sin embargo, presentan un grave problema: su velocidad está limitada por las características de la red.



De hecho, en el mejor de los casos sólo se pueden transmitir 9600 bits por segundo (bps), lo cual es muy lento. Esta limitación de velocidad es debida a que la red de telefonía está diseñada para transmitir señales analógicas que representan la voz y que tienen unas características particulares; mientras que en la comunicación de ordenadores necesitamos transmitir señales digitales, que tienen características distintas.



Para poder transmitir datos por red telefónica, es necesario hacer una conversión de las señales binarias (utilizadas por el ordenador) a señales analógicas (las transmitidas por la red telefónica).

La finalidad de cualquier tipo de red de comunicaciones es la de transferir información entre dos puntos. El punto desde el que se hace la transferencia de datos puede ser: un terminal, un ordenador, un teléfono, etc. Los puntos de la red donde se procesa la información reciben el nombre de nodos.

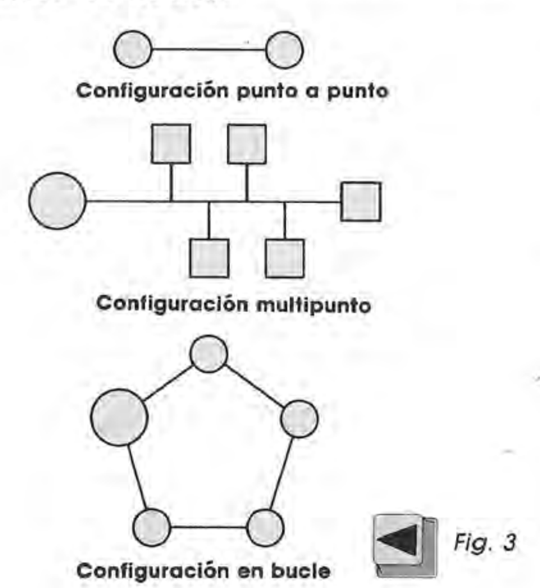


Fig. 3

La disposición geométrica de los nodos que forman la red determina su configuración. La figura 3 muestra algunos tipos de estructuras, donde los distintos nodos pueden representar terminales de ordenador, microordenadores conectados a un miniordenador o miniordenadores conectados a un ordenador.

Las configuraciones básicas utilizadas son: la red centralizada, la red distribuida y la red jerárquica o en árbol. Combinando convenientemente estos tipos básicos se pueden realizar otros tipos mucho más complejos.

Red centralizada. También se conocen a este tipo de redes con el nombre de Estrella, porque los diferentes nodos están conectados a través de un equipo central; todas las rutas del conjunto de nodos que rodea al ordenador central reciben el nombre de agrupación. Las distancias entre estos nodos que constituyen la agrupación no tiene porqué ser igual; de hecho suelen ser muy diferentes entre sí, dependiendo de la realización práctica de la red.

En una red de este tipo la potencia del equipo central supera a la de los ordenadores periféricos. Por esta razón el mayor inconveniente de esta distribución es su sensibilidad a los fallos del equipo central, ya que de producirse una avería en el nodo central se interrumpe todo el proceso de comunicación.
Un ejemplo típico de este tipo de configuración puede ser un ordenador central conectado a varios terminales.

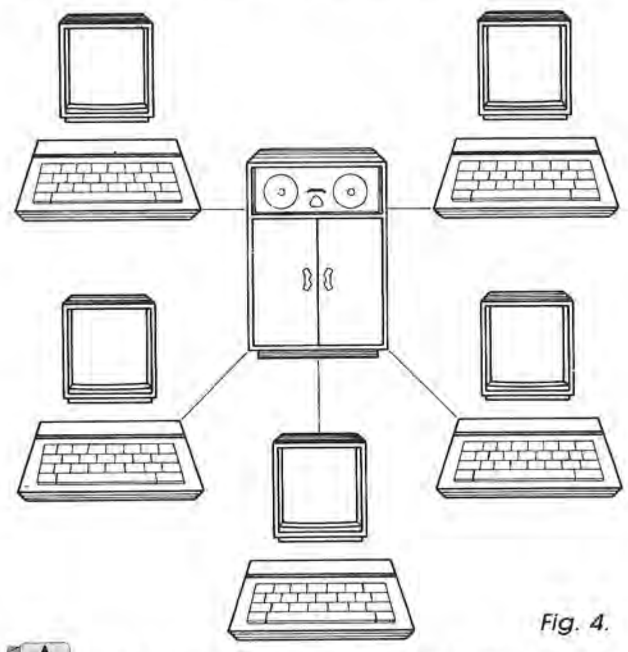
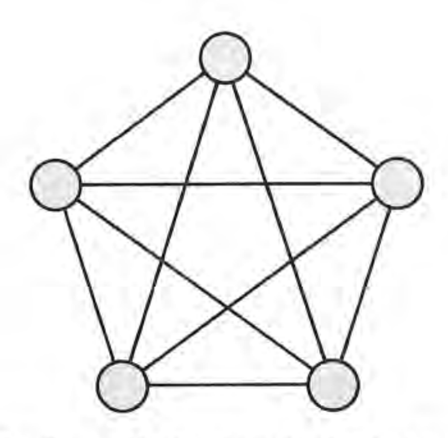


Fig. 4.

Configuración en estrella. Red centralizada.

Red distribuida. Este es el caso contrario al de las redes centralizadas; es decir, no existe ningún ordenador central y, por tanto, todos los ordenadores se reparten la responsabilidad de las comunicaciones; un mensaje puede pasar a través de varios miembros de la red antes de alcanzar su destino final. Lo normal es que cada equipo esté conectado, al menos, con otros dos más para que la comunicación se pueda realizar por caminos alternativos cuando una línea entre dos caminos esté averiada. Incluso si toda la línea se avería, los miembros no afectados pueden continuar trabajando mientras exista una línea en funcionamiento.

Cuando todos los ordenadores estén conectados entre sí, de forma que existe una conexión entre cada nodo, la configuración de la red se llama TOTALMENTE DISTRIBUIDA.



Configuración totalmente distribuida.

Como ejemplo de este tipo de red está la comunicación telefónica en una gran ciudad. Si surge algún problema en cualquiera de las líneas interurbanas, el tráfico se puede desviar por rutas alternativas.

Las principales ventajas que ofrecen estas redes es la flexibilidad y velocidad de respuesta, además de facilitar la transmisión de los datos y la compartición de los recursos.

Red jerárquica o en árbol. Una red jerárquica o en árbol tiende a hacer converger el flujo de datos desde el ordenador menos potente al de mayor capacidad. Su estructura tiene la forma de un árbol visto desde abajo.

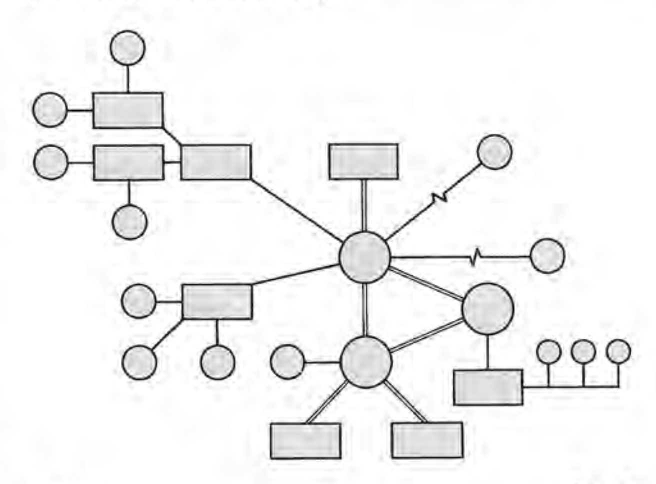
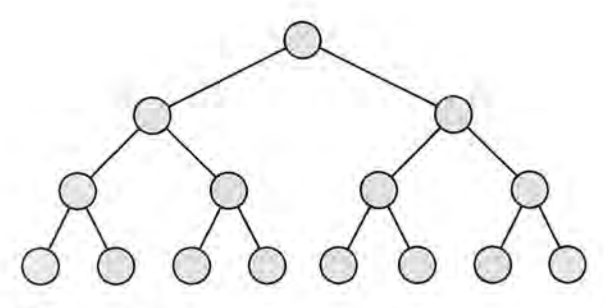


Fig. 5.

Red distribuida. En este caso ningún nodo centraliza el control.

Este tipo de red es muy utilizado en aplicaciones industriales parq supervisar y controlar varios procesos. Cada nivel del árbol ejecuta una tarea determinada





Red en árbol o jerárquica.

e informa a un nivel de supervisión superior. Los nodos de nivel inferior pueden ser sensores acoplados a microordenadores, los cuales están acoplados a miniordenadores, y éstos a su vez a uno o varios ordenadores.



Razones de implantación de una red

Aparte de las funciones específicas para las cuales haya sido diseñada una red, existen una serie de posibilidades que justifican la implantación de una red de ordenadores.

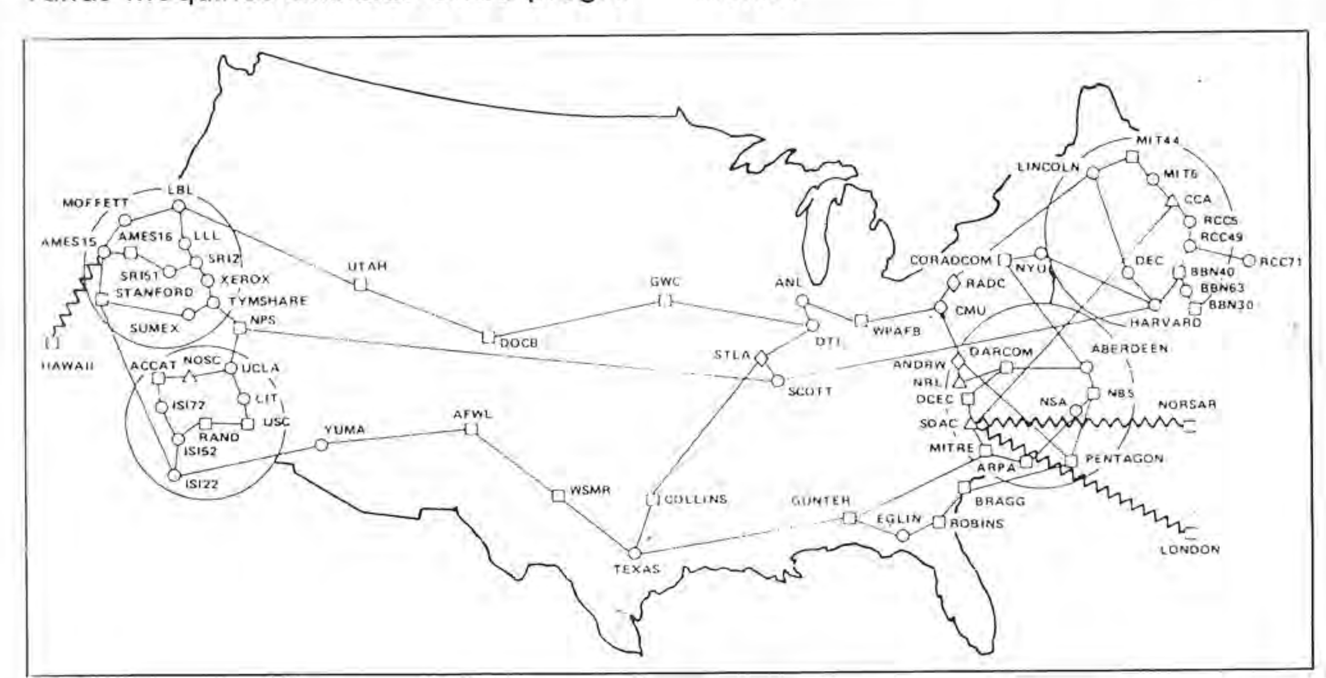
1. *Reparto de cargas.* Ejecutando en varias máquinas similares cada programa de una aplicación, se consigue tener el mismo rendimiento sin sobrecargar ningún ordenador. Esto se suele utilizar en la puesta a punto de programas.

2. *Eliminación de datos duplicados.* En una red es posible acceder a los datos almacenados en un ordenador desde otro cualquiera, lo que elimina los costes de duplicación de los distintos ficheros.

3. *Flexibilidad.* En muchas organizaciones se ha obligado a tener el mismo hardware y software con vistas a una posible "reprogramación"; si no se hubieran tomado estas medidas, la tarea de reorganización de cada aplicación podría ser enorme. La implantación de una red optimizaría aún más esta solución.

4. *Reducción de los costes de comunicación.* La sustitución de canales de baja velocidad por un circuito de alta velocidad, puede producir una considerable reducción del costo.

5. *Posibilidad de combinar los medios disponibles.* Con una red es posible combinar las diferentes máquinas instaladas y conseguir un sistema con posibilidades mucho mayores que las que ofrecía la suma de las capacidades de cada uno de los componentes trabajando aisladamente.





Configuración geográfica que tenía la red ARPANET en junio de 1980.

MAQUINA 6502

COMMODORE 64

Comandos o nemotécnicos utilizados por el microprocesador



OMO ya se ha visto, el microprocesador del COMMODORE-64 tiene seis registros propios, uno de los cuales, el registro "contador de programa", puede direccionar 2^{16}=65536 posiciones de memoria.

Por eso se dice que tiene un bus de direcciones de 16 bits. Este valor de 16 bits es la dirección del próximo comando que el microprocesador deberá buscar en la memoria y ejecutar.

También se dice que tiene un bus de datos de 8 bits, por lo que podrían existir 2^8=256 comandos diferentes.

Para aquél que no le quede muy claro la palabra "bus", diremos que se trata de unas líneas eléctricas que enlazan el microprocesador con la memoria, ya sea ROM o RAM, de esta manera se consigue una intercomunicación entre ambas partes.

Veamos ahora los comandos que vamos a utilizar. De las 256 posibilidades para obtener un comando, sólo 151 constituyen una instrucción válida para el microprocesador, y de éstas 151 instrucciones hay tan sólo 56 verdaderamente diferentes, lo que ocurre es que para cada una de ellas se pueden dar uno o varios modos de direccionamiento.

Vamos a presentar este juego de comandos y sus modos de direccionamiento por grupos.

Comandos de carga

Sirven para cargar datos desde la memoria hasta un registro del procesador. Como éste tiene tres registros de trabajo, debe haber tres comandos de carga:

LDA Carga al acumulador o ACU.
LDX Carga al registro X.
LDY Carga al registro Y.

Ahora bien, *¿cómo podemos cargar un número a cada uno de estos registros?* Mediante los modos de direccionamiento.



Modos de direccionamiento

1. Directamente y expresado mediante el símbolo (#) doble cruz. Esta forma de carga se denomina "Direccionamiento inmediato" y su equivalente en BASIC es la instrucción, A=5, X=7, Y=9.

LDA # $05 Carga el ACU con el valor $05.

LDX # $07 Carga el registro X con el valor $07.

LDY # $09 Carga el registro Y con el valor $09.

Imaginemos la instrucción LDA#$05. En la memoria, el comando y el valor se almacenan consecutivamente uno detrás del otro, por lo que si se arranca un programa en esta posición, el microprocesador extraería el contenido de la primera dirección y lo interpretaría como comando.

A continuación saltaría a la siguiente posición para cargar en el registro ACU el valor correspondiente 05. Según esto, se trata de un comando de dos bytes; por ello, el contador de programa (PC) se incrementa en dos posiciones, e indi-

ca el próximo comando que puede ser interpretado por el microprocesador.

2. A través del contenido de una determinada posición de memoria, y no necesariamente con un valor constante como en el direccionamiento inmediato. A esta forma de carga se le denomina direccionamiento absoluto:

LDA $ C000	Carga el ACU con el valor contenido en la posición $ C000 en hexadecimal.
LDX $ 0801	Carga el registro X con el valor contenido en la posición $ 0801.
LDY $ 8000	Carga el registro Y con el valor contenido en la posición $ 8000.

Aquí puede surgir una duda: ¿cómo colocar el número $ XXXX en general, de 16 bits en un registro o posición de memoria que sólo tiene 8 bits?

Nada más sencillo; descomponiendo dicho número en dos partes, que se denominan byte alto (HB) y byte bajo (LB), y almacenando primeramente la parte baja y a continuación la parte alta.

Veámoslo con un ejemplo:

LDA $ C000 $ C000 = 49152

Primero se almacenaría el código del comando LDA para este modo de direccionamiento, que es $AD=173, a continuación la porción baja de la dirección $00=0 y, finalmente, la porción alta $CO=192.

Así, pues, quedaría la serie $AD, $00, $C0, o bien, 173, 0, 192.

La correspondiente orden BASIC es A=PEEK ($C000).

Se observa que este comando es de tres bytes, uno para el comando propiamente dicho y dos más para la posición de memoria de donde debe ser extraído el dato.

Antes de continuar vamos a echar un vistazo a otro de los registros del microprocesador: el registro de estado, el cual nos hace saber del último comando ejecutado por el microprocesador, de tal manera que consultándole puede servirnos de base para la toma de decisiones según los cambios sufridos en él, así como para la utilización de los comandos condicionales.

En este registro cada bit tiene un nombre y un significado específico denominándose comúnmente flags. Cada uno de estos flags puede estar a "cero" o a "uno".

REGISTRO DE ESTADO
7 6 5 4 3 2 1 0
N V - B D I Z C

Carry-C indica si en una operación se ha producido un desbordamiento, es decir, si en una suma se obtiene un número superior a 255=$FF, se activa el flag.

Así, $FC(252) + $08(8) = $05(5) y el carry flag activado indicando el desbordamiento.

Zero-Z se activa (=1) cuando el resultado de una operación es cero.

Interrupt-I. Este flag determina si están o no permitidas las interrupciones en un programa. Lo estudiaremos mejor más adelante.

Decimal-D. Determina si en una adición o sustracción se está trabajando en forma binaria (desactivado) o en forma decimal (activado).

Break-B. Indica una interrupción debida al comando BRK, que veremos cuando hablemos del manejo de interrupciones.

Overflow-V. Este flag sólo se utiliza cuando trabajamos con números con signo para indicar un desbordamiento.

Negative-N. Este flag se activa siempre que el resultado de una operación sea mayor que 127, lo cual quiere decir que el séptimo bit del registro en cuestión está activado.

Ahora ya podemos volver donde nos quedamos y decir que con cada comando de carga se incide sobre los flágs Zero y Negative dependiendo del valor cargado.

3. A través del contenido de una posición de memoria en el rango de 0 a 255, y es lo que se llama "direccionamiento Zero page" o página cero. De esta forma sólo necesitaremos dos bytes para expresar el comando completo.

El primero para el código del comando y el segundo para la posición de memoria de donde ha de extraerse el dato.

Este modo de direccionamiento es una particularidad de los procesadores de la serie 65XX, y es una gran ventaja, ya que en las primeras 256 posiciones de memoria se encuentran la mayoría de las variables del sistema que pueden ser modificadas o consultadas mediante un comando de tan sólo dos bytes.

LDA $ 2B A = PEEK($2B)



PROGRAMAS

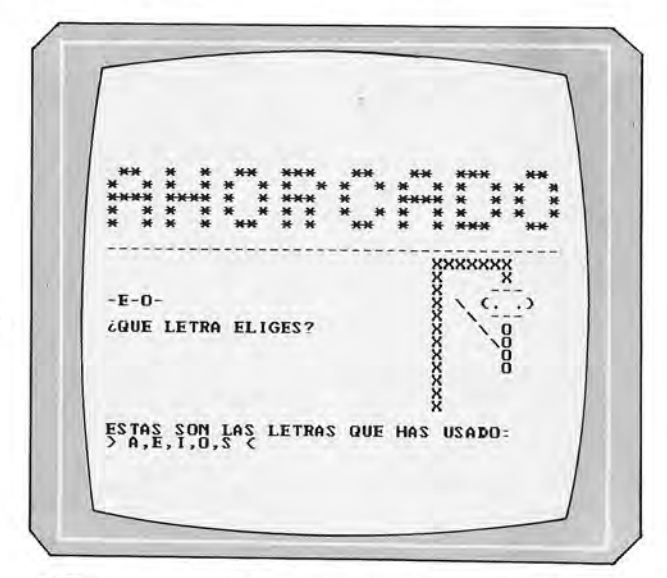
EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS



Programa: El Ahorcado

El primer programa de este octavo tomo nos permitirá jugar al famoso juego de «El Ahorcado». Poco hay que decir sobre las reglas del mismo, ya que son por todos conocidas. No obstante, y de forma resumida, las vamos a dar a continuación.

El juego consiste en adivinar una palabra que ha pensado el ordenador. Para ello vamos diciéndole letras. Si acertamos una letra, el ordenador la pondrá en su lugar dentro de la palabra. Si dicha letra estuviese repetida más de una vez en la palabra, el ordenador nos imprimirá cada vez que ésta aparezca en su lugar. Cada vez que fallemos el ordenador nos dibujará una parte del cuerpo de un ahorcado. El juego termina cuando el cuerpo está completo y, por tanto, estás muerto.





Pantalla del programa «El Ahorcado» en plena ejecución.

```
100 REM  ***********************************************************************
101 REM  *                                                                     *
102 REM  *   EEEEE L         AAA  H   H  OOO  RRRR   CCC   AAA  DDDD   OOO     *
103 REM  *   E     L        A   A H   H O   O R   R C   C A   A D   D O   O    *
104 REM  *   EEE   L        AAAAA HHHHH O   O RRRR  C     AAAAA D   D O   O    *
105 REM  *   E     L        A   A H   H O   O R R   C   C A   A D   D O   O    *
106 REM  *   EEEEE LLLLL    A   A H   H  OOO  R   R  CCC  A   A DDDD   OOO     *
107 REM  *                                                                     *
108 REM  ***********************************************************************
```

```
109 REM
110 REM  ******************************************************
111 REM  ******************************************************
112 REM  *************** (c) Ed. Siglo Cultural ***************
113 REM  *************** (c) 1987.              ***************
114 REM  ******************************************************
115 REM  ******************************************************
116 REM
117 REM *** INICIALIZACION DEL PROGRAMA ***
118 REM
119 CLS
120 PRINT" **  *  *  **  ***   **   **  ***   **"
121 PRINT"*  * *  * *  * *  * *  * *  * *  * *  *"
122 PRINT"**** **** *  * ***  *    **** *  * *  *"
123 PRINT"*  * *  * *  * * *  *  * *  * *  * *  *"
124 PRINT"*  * *  *  **  * *   **  *  * ***   **"
125 PRINT
126 PRINT"---------------------------------------"
127 DIM P$(12,12):DIM L$(20):DIM D$(20):DIM N$(27)
128 LET C=1
129 LET N=77
130 DIM U(N)
131 FOR I=1 TO 20
132    LET D$(I)="-"
133 NEXT I
134 LET M=0
135 FOR I=1 TO 26
136    LET N$(I)=""
137 NEXT I
138 FOR I=1 TO 12
139    FOR J=1 TO 12
140       LET P$(I,J)=" "
141    NEXT J
142 NEXT I
143 FOR I=1 TO 12
144    LET P$(I,1)="X"
145 NEXT I
146 FOR I=1 TO 7
147    LET P$(1,I)="X"
148 NEXT I
149 LET P$(2,7)="X"
150 REM
151 REM *** PROGRAMA PRINCIPAL ***
152 REM
153 FOR I=8 TO 23
154    LOCATE I,1
155    PRINT SPACE$(40);
156 NEXT I
157 IF C<N THEN GOTO 160
158 LOCATE 21,1
159 PRINT "HAS ADIVINADO TODAS LAS PALABRAS!":GOTO 406
160 LET Q=INT(N*RND)+1
161 IF U(Q)=1 THEN GOTO 160
162 LET U(Q)=1
163 LET C=C+1
164 RESTORE
165 LET T1=0
166 FOR I=1 TO Q
167    READ A$
168 NEXT I
169 LET L=LEN(A$)
170 FOR I=1 TO L
171    LET L$(I)=MID$(A$,I,1)
172 NEXT I
173 GOTO 303
174 LOCATE 21,1
175 PRINT  "ESTAS SON LAS LETRAS QUE HAS USADO:
176 LOCATE 22,1
177 PRINT SPACE$(40)
```

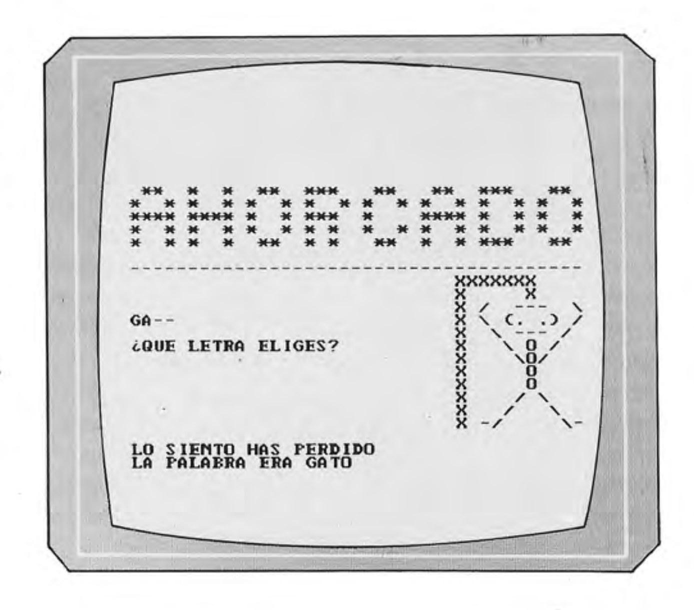
```
178 LOCATE 22,1
179 PRINT "> ";
180 FOR I=1 TO 26
181     PRINT N$(I);
182     IF N$(I+1)="" THEN GOTO 185
183     PRINT ",";
184 NEXT I
185 PRINT " <"
186 LOCATE 11,1
187 FOR I=1 TO L
188    PRINT D$(I);
189 NEXT I
190 LOCATE 13,1
191 PRINT "(QUE LETRA ELIGES?"
192 LET G$=INKEY$
193 IF (G$>"Z" AND G$<"A") AND (G$>"z" OR G$<"a") OR G$="" THEN GOTO 192
194 LET R=0
195 FOR I=1 TO 26
196    IF N$(I)="" THEN GOTO 199
197    IF G$=N$(I) THEN LOCATE 21,1:PRINT "ESA YA LA HAS ELEGIDO
  ":LOCATE 22,1:PRINT SPACE$(40):FOR I=1 TO 1000:NEXT I:GOTO 174
198 NEXT I
199 LET N$(I)=G$
200 LET T1=T1+1
201 FOR I=1 TO L
202    IF L$(I)=G$ GOTO 206
203 NEXT I
204 IF R=0 THEN GOTO 209
205 GOTO 211
206 LET D$(I)=G$
207 LET R=R+1
208 GOTO 203
209 LET M=M+1
210 GOTO 241
211 FOR I=1 TO L
212    IF D$(I)="-" THEN GOTO 215
213 NEXT I
214 GOTO 236
215 LOCATE 11,1
216 FOR I=1 TO L
217    PRINT D$(I);
218 NEXT I
219 LOCATE 21,1
220 PRINT SPACE$(80)
221 LOCATE 21,1
222 INPUT "(QUE PALABRA ELIGES";B$
223 IF A$=B$ THEN GOTO 229
224 LOCATE 21,1
225 PRINT "NO. PRUEBA CON OTRA LETRA.":PRINT SPACE$(40)
226 FOR I=1 TO 1000
227 NEXT I
228 GOTO 174
229 LOCATE 21,1:PRINT "LO HAS HECHO EN"; T1;"INTENTOS."
230 PRINT "(QUIERES OTRA PALABRA? (S/N)"
231 LET W$=INKEY$
232 IF W$="S" OR W$="s" THEN GOTO 131
233 IF W$="N" OR W$="n" THEN GOTO 406
234 GOTO 231
235 GOTO 406
236 LOCATE 21,1
237 PRINT SPACE$(80)
238 LOCATE 21,1
239 PRINT "HAS ENCONTRADO LA PALABRA"
240 GOTO 230
241 LOCATE 21,1
242 PRINT "ESA LETRA NO ESTA EN LA PALABRA        "
243 ON M GOTO 244,246,248,250,252,254,256,258,260,262
244 PRINT "PRIMERO PINTO UNA CABEZA               "
245 GOTO 263
```

```
246 PRINT "AHORA PINTO UN CUERPO                              "
247 GOTO 263
248 PRINT "LO SIGUIENTE QUE DIBUJO ES UN BRAZO                "
249 GOTO 263
250 PRINT "ES HORA DE PONER EL OTRO BRAZO                     "
251 GOTO 263
252 PRINT  "AHORA LE PONEMOS LA PIERNA DERECHA                "
253 GOTO 263
254 PRINT "ESTA VEZ LE PONGO LA PIERNA IZQUIERDA              "
255 GOTO 263
256 PRINT "AHORA UNA MANO                                     "
257 GOTO 263
258 PRINT "LO SIGUIENTE LA OTRA MANO                          "
259 GOTO 263
260 PRINT "AHORA LE DIBUJAMOS UN PIE                          "
261 GOTO 263
262 PRINT "AQUI ESTA EL OTRO PIE. ESTAS AHORCADO!! "
263 ON M GOTO 264,275,279,283,288,291,294,296,298,301
264 LET P$(3,6)="-"
265 LET P$(3,7)="-"
266 LET P$(3,8)="-"
267 LET P$(4,5)="("
268 LET P$(4,6)="."
269 LET P$(4,8)="."
270 LET P$(4,9)=")"
271 LET P$(5,6)="-"
272 LET P$(5,7)="-"
273 LET P$(5,8)="-"
274 GOTO 303
275 FOR I=6 TO 9
276    LET P$(I,7)="O"
277 NEXT I
278 GOTO 303
279 FOR I=4 TO 7
280    LET P$(I,I-1)="\"
281 NEXT I
282 GOTO 303
283 LET P$(4,11)="/"
284 LET P$(5,10)="/"
285 LET P$(6,9)="/"
286 LET P$(7,8)="/"
287 GOTO 303
288 LET P$(10,6)="/"
289 LET P$(11,5)="/"
290 GOTO 303
291 LET P$(10,8)="\"
292 LET P$(11,9)="\"
293 GOTO 303
294 LET P$(3,11)="\"
295 GOTO 303
296 LET P$(3,3)="/"
297 GOTO 303
298 LET P$(12,10)="\"
299 LET P$(12,11)="-"
300 GOTO 303
301 LET P$(12,3)="-"
302 LET P$(12,4)="/"
303 FOR I=1 TO 12
304    LOCATE 7+I,29
305    FOR J=1 TO 12
306       PRINT P$(I,J);
307    NEXT J
308 NEXT I
309 LOCATE 21,1:PRINT SPACE$(80)
310 LOCATE 21,1
311 IF M<>10 THEN FOR I=1 TO 1000:NEXT I:GOTO 174
312 PRINT "LO SIENTO HAS PERDIDO":PRINT "LA PALABRA ERA ";A$
313 FOR I=1 TO 2000:NEXT I
314 LOCATE 21,1
```

```
315 PRINT SPACE$(80)
316 GOTO 230
317 DATA "CAL"
318 DATA "TOS"
319 DATA "SON"
320 DATA "PAN"
321 DATA "POR"
322 DATA "SIN"
323 DATA "FEO"
324 DATA "MES"
325 DATA "DOS"
326 DATA "CAN"
327 DATA "GATO"
328 DATA "MESA"
329 DATA "BOTE"
330 DATA "BACA"
331 DATA "BOLI"
332 DATA "CARA"
333 DATA "TRES"
334 DATA "PINO"
335 DATA "PALO"
336 DATA "CASO"
337 DATA "PIZZA"
338 DATA "RELOJ"
339 DATA "FICHA"
340 DATA "TECLA"
341 DATA "ACIDO"
342 DATA "FALTA"
343 DATA "CINCO"
344 DATA "SIETE"
345 DATA "MORRO"
346 DATA "PELAS"
347 DATA "SALTAR"
348 DATA "DINERO"
349 DATA "ARRIBA"
350 DATA "SEMANA"
351 DATA "HUMEDO"
352 DATA "GAKEON"
353 DATA "PENSAR"
354 DATA "PROBAR"
355 DATA "CONTRA"
356 DATA "PERICO"
357 DATA "GESTION"
358 DATA "REPLICA"
359 DATA "MECHERO"
360 DATA "CIGARRO"
361 DATA "DRACULA"
362 DATA "OCTUBRE"
363 DATA "VAMPIRO"
364 DATA "DESTACA"
365 DATA "RESUMIR"
366 DATA "CONTROL"
367 DATA "SARGENTO"
368 DATA "CACHARRO"
369 DATA "TELEFONO"
370 DATA "DORMILON"
371 DATA "ALMOHADA"
372 DATA "RESUMIDO"
373 DATA "PERDIDOS"
374 DATA "ANIMALES"
375 DATA "COMPONER"
376 DATA "REPLICAR"
377 DATA "TRIANGULO"
378 DATA "PELIGROSO"
379 DATA "FABRICADO"
380 DATA "DIFERENTE"
381 DATA "ROTULADOR"
382 DATA "ESPEJISMO"
383 DATA "DEMOCRATA"
```

```
384 DATA "ESPARRAGO"
385 DATA "ABOLICION"
386 DATA "PERDICION"
387 DATA "ENFERMEDAD"
388 DATA "CIENTIFICO"
389 DATA "CALENDARIO"
390 DATA "IMPREGNADO"
391 DATA "DESTROZADO"
392 DATA "PATIDIFUSO"
393 DATA "IZQUIERDAS"
394 DATA "CARACTERES"
395 DATA "EVIDENCIAR"
396 DATA "DESARROLLO"
397 DATA "PARASIMPATICOMIMETICO"
398 DATA "PREMATRIMONIAL"
399 DATA "QUIMIOTROPISMO"
400 DATA "ESTERNOCLEIDO"
401 DATA "INSTITUCIONALIZACION"
402 DATA "CONSTITUCIONAL"
403 DATA "DESCENTRALIZACION"
404 PRINT  "ADIOS"
405 END
406 LOCATE 22,1:PRINT SPACE$(80)
407 LOCATE 22,1
408 PRINT "JUGAMOS OTRA VEZ? (S/N)"
409 LET A$=INKEY$
410 IF A$="S" OR A$="s" THEN RUN
411 IF A$="N" OR A$="n" THEN GOTO 413
412 GOTO 409
413 CLS
414 PRINT "HA SIDO ESTUPENDO JUGAR."
415 PRINT
416 PRINT "HASTA PRONTO."
417 PRINT:PRINT
418 END
```

Este programa funciona perfectamente en IBM; para que funcione en el COMMODORE, AMSTRAD y MSX habrá que realizar los cambios que aparecen más adelante. Para los usuarios del SPECTRUM, este programa volverá a aparecer en tomos sucesivos.





Cuando el cuerpo del ahorcado está completo, el juego termina.

COMMODORE:

```
119 PRINT CHR$(147)
154 POKE 214,I-1:POKE 211,0
155 FOR J=1 TO 40:PRINT " ";:NEXT J
158 POKE 214,20:POKE 211,0
160 LET Q=INT(N*RND(0))+1
174 POKE 214,20:POKE 211,0
176 POKE 214,21:POKE 211,0
177 FOR J=1 TO 40:PRINT " ";:NEXT J
186 POKE 214,10:POKE 211,0
190 POKE 214,12:POKE 211,0
192 GET G$
197 IF G$=N$(I) THEN POKE 214,20:POKE
211,0:PRINT "ESA YA LA HAS ELEGIDO
        ":POKE 214,22:POKE 211,0:FOR
J=1 TO 40:PRINT " ";:NEXT J:FOR I=1 TO
1000:NEXT I:GOTO 174
215 POKE 214,10:POKE 211,0
219 POKE 214,20:POKE 211,0
220 FOR J=1 TO 80:PRINT " ";:NEXT J
221 POKE 214,20:POKE 211,0
224 POKE 214,20:POKE 211,0
225 PRINT "NO. PRUEBA CON OTRA LE-
TRA":FOR J=1 TO 40:PRINT " ";:NEXT J
229 POKE 214,20:POKE 211,0:PRINT "LO
HAS HECHO EN";T1;"INTENTOS."
231 GET W$
236 POKE 214,20:POKE 211,0
237 FOR J=1 TO 80:PRINT " ";:NEXT J
238 POKE 214,20:POKE 211,0
241 POKE 214,20:POKE 211,0
304 POKE 214,6+I:POKE 211,28
309 POKE 214,20:POKE 211,0:FOR J=1 TO
80:PRINT " ";:NEXT J
310 POKE 214,20:POKE 211,0
315 FOR J=1 TO 80:PRINT " ";:NEXT J
406 POKE 214,21:POKE 211,1:FOR J=1 TO
80:PRINT " ";:NEXT J
407 POKE 214,21:POKE 211,0
409 GET A$
413 PRINT CHR$(147)
```

AMSTRAD:

```
154 LOCATE 1,I
158 LOCATE 1,21
174 LOCATE 1,21
176 LOCATE 1,22
178 LOCATE 1,22
186 LOCATE 1,11
190 LOCATE 1,13
197 IF G$=N$(I) THEN LOCATE 1,21:PRINT
"ESA YA LA HAS ELEGIDO":LOCATE 1,22:PRINT
SPACE$(40):FOR I=1 TO 1000:NEXT I:GOTO
174
215 LOCATE 1,11
219 LOCATE 1,21
221 LOCATE 1,21
224 LOCATE 1,21
229 LOCATE 1,21:PRINT "LO HAS HECHO
EN";T1;"INTENTOS."
236 LOCATE 1,21
238 LOCATE 1,21
241 LOCATE 1,21
304 LOCATE 29,7+I
309 LOCATE 1,21:PRINT SPACE$(80)
310 LOCATE 1,21
314 LOCATE 1,21
406 LOCATE 1,22:PRINT SPACE$(80)
407 LOCATE 1,22
```

MSX:

Las variaones que hay que hacer para el MSX son las mismas que para el AMSTRAD, pero cambiando también la línea 160 por:

```
160 LET Q=INT(N*RND(0))+1
```



Programa: Trivial ríos

El programa que aparece a continuación nos permitirá repasar nuestros conocimientos de los ríos más importantes de España. La forma que tiene el ordenador de preguntar no es la normal, sino que nos pregunta a base de iconos por pantalla.

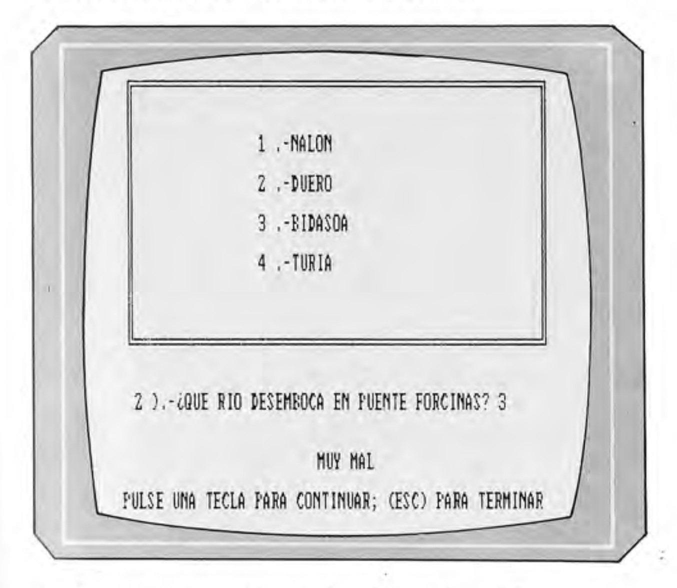



Pantalla de presentación del programa «Trivial Ríos».

Las preguntas van referidas, principalmente, al entorno en el cual están los ríos. Esto es, el ordenador nos va a hacer preguntas como:

¿Sabes qué río pasa por Soria?

De esta manera se consigue que el programa resulte más ameno y menos aburrido para los estudiantes.

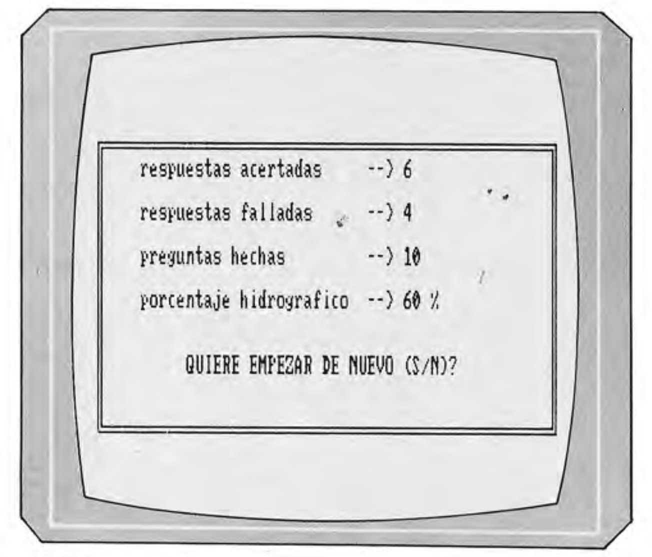




Un momento en la ejecucón del programa.

Este programa sólo es válido para el IBM. Las diferentes versiones para los demás ordenadores irán apareciendo en tomos sucesivos.

Por otro lado, al final del programa se imprime el número de preguntas hechas, el número de respuestas que han sido correctas, el tanto por ciento de aciertos (porcentaje hidrográfico).





Al final del programa, se dan los resultados que ha obtenido el usuario.

```
 1 REM  ************************************************************************
 2 REM  ***********        RIOS DE ESPA%A (TRIVIAL RIOS)       *****************
 3 REM  ***********     POR:    JUAN MANUEL GUTIERREZ LEITON   *****************
 4 REM  ***********                                            *****************
 5 REM  ************************************************************************
 6 REM
 7 REM  ************************************************************************
 8 REM  ************** (C) EDICIONES SIGLO CULTURAL (1987) ******************
 9 REM  ************************************************************************
10 DIM TR(75)
11 DIM AL(3)
12 DIM RI$(12)
13 RESTORE
14 CLS
15 PRINT
16 PRINT
17 PRINT"      222222222  222222222  222  22         22  222  22222222  222     "
18 PRINT"         222     22     22        22       22        22     22  222     "
19 PRINT"         222     222222222  222    22     22    222  22     22  222     "
20 PRINT"         222     22  222    222     22   22     222  22222222  222     "
21 PRINT"         222     22   222   222      22 22      222  22     22  222     "
22 PRINT"         222     22    222  222       222       222  22     22  2222222"
23 PRINT
24 PRINT
25 PRINT
26 PRINT"                222222222     222    222222222    222222222            "
27 PRINT"                22     22            22     22    22                   "
28 PRINT"                222222222     222    22     22    222222222            "
29 PRINT"                22  222       222    22     22           22            "
30 PRINT"                22   222      222    22     22           22            "
31 PRINT"                22    222     222    222222222    222222222            "
32 REM contadores de las respuestas acertadas y equivocadas
```

```
33 BI=0
34 MA=0
35 FOR I=1 TO 25
36   RANDOMIZE TIMER
37   SOUND RND*100+37,2
38 NEXT I
39 LOCATE 22,23
40 PRINT "PULSE UNA TECLA PARA CONTINUAR"
41 LET A$=INKEY$
42 IF A$="" THEN 41
43 LOCATE 22,23
44 PRINT "CARGANDO DATOS,POR FAVOR ESPERE"
45 REM carga de los datos de los rios
46 GOSUB  128
47 REM carga de los datos de la tabla de respuestas
48 GOSUB  223
49 REM el siguiente bucle for se ejecuta tantas veces como preguntas hay
50 REM en este caso hay 54 preguntas
51 FOR I=1 TO 54
52 REM ir a la subrutina de dibujo de la ventana
53   GOSUB 284
54   READ PR$
55   LET RE=TR(I)
56   RANDOMIZE TIMER
57   LET AL(1)=INT(RND*12)+1
58   IF AL(1)=RE THEN GOTO 56
59   RANDOMIZE TIMER
60   LET AL(2)=INT(RND*12)+1
61   IF AL(2)=AL(1) OR AL(2)=RE THEN GOTO 59
62   RANDOMIZE TIMER
63   LET AL(3)=INT(RND*12)+1
64   IF AL(3)=AL(2) OR AL(3)=AL(1) OR AL(3)=RE THEN GOTO 62
65   LET PO=INT(RND*4)+1
66   REM po es la posicion en la ventana de la respuesta
67   LOCATE 3+PO*2,30
68   PRINT PO;".-";RI$(RE)
69   LET K=1
70   FOR J=1 TO 3
71     IF PO=J THEN GOTO 73
72     GOTO 77
73     LET K=K+1
74     LOCATE 3+K*2,30
75     PRINT K;".-";RI$(AL(J))
76     GOTO 79
77     LOCATE 3+K*2,30
78     PRINT K;".-";RI$(AL(J))
79     LET K=K+1
80   NEXT J
81   LOCATE 18,15
82   PRINT I;").-";PR$;
83   INPUT CO$
84   REM co$ es la contestacion dada a una pregunta
85   IF LEN(CO$)>1 THEN BEEP:GOTO 81
86   IF CO$<"0" OR CO$>"9" THEN BEEP:GOTO 81
87   IF VAL(CO$)<1 OR VAL(CO$)>4 THEN BEEP:GOTO  81
88   IF VAL(CO$)=PO THEN GOTO 97
89   LOCATE 21,35
90   SOUND 100,10
91   PRINT "   MUY MAL   "
92   MA=MA+1
93   COLOR 31
94   LOCATE 3+PO*2,30
95   PRINT PO;".-";RI$(RE)
96   GOTO 101
97   LOCATE 21,35
98   SOUND 1000,5
99   PRINT "   MUY BIEN   "
100  BI=BI+1
```

```
101   COLOR 2
102   LOCATE 23,17
103   PRINT "PULSE UNA TECLA PARA CONTINUAR; (ESC) PARA TERMINAR"
104   A$=INKEY$
105   IF A$="" THEN 104
106   IF ASC(A$)=27 THEN GOTO 108
107 NEXT I
108 GOSUB 287
109 LOCATE 3,20
110 PRINT "respuestas acertadas     -->";BI
111 LOCATE 5,20
112 PRINT "respuestas falladas      -->";MA
113 LOCATE 7,20
114 PRINT "preguntas hechas         -->";BI+MA
115 LOCATE 9,20
116 COLOR 31
117 PRINT "porcentaje hidrografico  -->";INT((BI/I)*100);"%"
118 COLOR 2
119 LOCATE 12,25
120 PRINT "QUIERE EMPEZAR DE NUEVO (S/N)?";
121 LET A$=INKEY$
122 IF A$="" THEN GOTO 121
123 IF A$="S" OR A$="s" THEN GOTO 13
124 CLS
125 END
126 REM  *****************************************************************
127 REM             data de los principales rios de la peninsula
128 REM  *****************************************************************
129 REM
130 DATA MI%O
131 DATA DUERO
132 DATA TAJO
133 DATA GUADIANA
134 DATA GUADALQUIVIR
135 DATA EBRO
136 DATA JUCAR
137 DATA SEGURA
138 DATA NALON
139 DATA NERVION
140 DATA BIDASOA
141 DATA TURIA
142 REM  *****************************************************************
143 REM             fin de la data de los rios de la peninsula
144 REM  *****************************************************************
145 REM
146 REM
147 REM  *****************************************************************
148 REM             creacion de la tabla de rios de la peninsula
149 REM  *****************************************************************
150 FOR I=1 TO 12
151   READ RI$(I)
152 NEXT I
153 RETURN
154 REM  *****************************************************************
155 REM           fin de la creacion de la tabla de rios
156 REM  *****************************************************************
157 REM
158 REM
159 REM  *****************************************************************
160 REM       data de las 54 preguntas de los rios de la peninsula
161 REM  *****************************************************************
162 DATA (QUE RIO PIRENAICO DESEMBOCA EN EL CANTABRICO
163 DATA (QUE RIO DESEMBOCA EN PUENTE FORCINAS
164 DATA (EL RIO CABRIEL ES UN AFLUENTE DEL...
165 DATA (EL RIO TAMBRE ES UN AFLUENTE DEL...
166 DATA (EL RIO MULA ES UN AFLUENTE DEL...
167 DATA (EL RIO GUADIANA MENOR ES UN AFLUENTE DEL...
168 DATA (QUE RIO DESEMBOCA EN OPORTO
169 DATA (QUE RIO PASA POR BURGOS
```

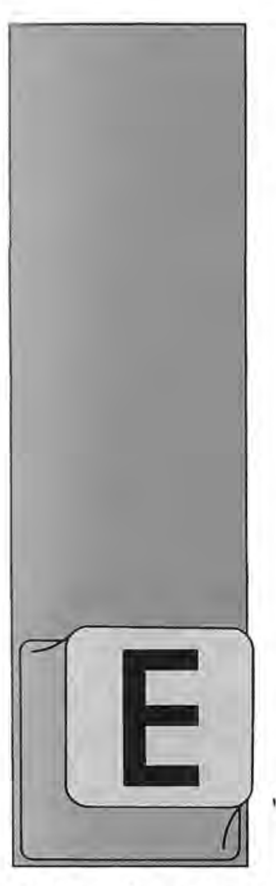
```
170 DATA (QUE RIO DESEMBOCA EN AYAMONTE (HUELVA)
171 DATA (QUE RIO DESEMBOCA EN LA GUARDIA
172 DATA (EL RIO GALLO ES UN AFLUENTE DEL...
173 DATA (EL RIO TORMES ES UN AFLUENTE DEL...
174 DATA (EL RIO ESLA ES UN AFLUENTE DEL...
175 DATA (QUE RIO PASA POR ORENSE
176 DATA (QUE RIO PASA POR LA CIUDAD DE TOLEDO
177 DATA (QUE RIO EN SU TRAMO FINAL SEPARA ESPA%A DE FRANCIA
178 DATA (QUE RIO NACE EN PE%A ORDU%A (ALAVA)
179 DATA (EL RIO NARCEA ES UN AFLUENTE DEL...
180 DATA (QUE RIO PASA POR ELIZONDO
181 DATA (EL RIO MUNDO ES UN AFLUENTE DEL...
182 DATA (QUE RIO DESEMBOCA EN CULLERA
183 DATA (QUE RIO PASA POR SORIA
184 DATA (EL RIO JALON ES UN AFLUENTE DEL...
185 DATA (EL RIO GENIL ES UN AFLUENTE DEL...
186 DATA (QUE RIO NACE ENTRE LAS SIERRAS DE ALCON Y CAZORLA
187 DATA (QUE RIO PASA POR CORDOBA Y SEVILLA
188 DATA (QUE RIO NACE EN FONTIBRE (SANTANDER)
189 DATA (QUE RIO PASA POR ZARAGOZA
190 DATA (QUE RIO DESEMBOCA EN SANLUCAR DE BARRAMEDA (HUELVA)
191 DATA (EL RIO CADAGUA ES UN AFLUENTE DEL...
192 DATA (QUE RIO NACE EN LA PROVINCIA DE ASTURIAS
193 DATA (QUE RIO PASA POR BILBAO
194 DATA (QUE RIO NACE JUNTO AL CERRO DE SAN FELIPE EN CUENCA
195 DATA (QUE RIO DESEMBOCA EN GUARDAMAR (ALICANTE)
196 DATA (QUE RIO PASA POR VALENCIA
197 DATA (QUE RIO NACE EN LA LAGUNA DE FUENMI%A (LUGO)
198 DATA (EL RIO SIL ES UN AFLUENTE DEL...
199 DATA (EL PISUERGA ES UN AFLUENTE DEL...
200 DATA (QUE RIO ES EL MAS LARGO DE LA PENINSULA (1.120 Km)
201 DATA (QUE RIO NACE EN LOS PICOS DE URBION
202 DATA (QUE RIO NACE EN LA SIERRA DE ALBARRACIN
203 DATA (QUE RIO PASA POR LA CIUDAD DE BADAJOZ
204 DATA (EL RIO GUADALBULLON ES UN AFLUENTE DEL...
205 DATA (EL RIO CHANZA ES UN AFLUENTE DEL...
206 DATA (EL RIO GENIL ES UN AFLUENTE DEL...
207 DATA (QUE RIO NACE EN FUENTE SEGURA (JAEN)
208 DATA (EL RIO TAIBILLA ES UN AFLUENTE DEL...
209 DATA (EL RIO HIJAR ES UN AFLUENTE DEL...
210 DATA (EL RIO GALLEGO ES UN AFLUENTE DEL...
211 DATA (QUE RIO DESEMBOCA EN LA CIUDAD DE LISBOA
212 DATA (EL RIO GUADIELA ES UN AFLUENTE DEL...
213 DATA (EL RIO ARAGON ES UN AFLUENTE DEL...
214 DATA (EL RIO IBAIZABAL ES UN AFLUENTE DEL...
215 DATA (QUE RIO NACE EN CASAS DE FUENTE GARCIA (TERUEL)
216 REM ********************************************************************
217 REM     fin de la data de las 54 preguntas de los rios
218 REM ********************************************************************
219 REM
220 REM
221 REM ********************************************************************
222 REM                 creacion de la tabla de respuestas
223 REM ********************************************************************
224 LET TR(1)=11 :REM a la pregunta numero 1 le corresponde el rio ri$(11)
225 LET TR(2)=9  :REM a la pregunta numero 2 le corresponde el rio ri$(9)
226 LET TR( 3)=7
227 LET TR( 4)=1
228 LET TR( 5)=8
229 LET TR( 6)=5
230 LET TR( 7)=2
231 LET TR( 8)=6
232 LET TR( 9)=4
233 LET TR(10)=1
234 LET TR(11)=3
235 LET TR(12)=2
236 LET TR(13)=2
237 LET TR(14)=1
```

```
238 LET TR(15)=3
239 LET TR(16)=11
240 LET TR(17)=10
241 LET TR(18)=9
242 LET TR(19)=11
243 LET TR(20)=8
244 LET TR(21)=7
245 LET TR(22)=2
246 LET TR(23)=6
247 LET TR(24)=4
248 LET TR(25)=5
249 LET TR(26)=5
250 LET TR(27)=6
251 LET TR(28)=6
252 LET TR(29)=5
253 LET TR(30)=10
254 LET TR(31)=9
255 LET TR(32)=10
256 LET TR(33)=7
257 LET TR(34)=8
258 LET TR(35)=7
259 LET TR(36)=1
260 LET TR(37)=1
261 LET TR(38)=2
262 LET TR(39)=3
263 LET TR(40)=2
264 LET TR(41)=3
265 LET TR(42)=4
266 LET TR(43)=5
267 LET TR(44)=4
268 LET TR(45)=5
269 LET TR(46)=8
270 LET TR(47)=8
271 LET TR(48)=6
272 LET TR(49)=6
273 LET TR(50)=3
274 LET TR(51)=3
275 LET TR(52)=6
276 LET TR(53)=10
277 LET TR(54)=3
278 RETURN
279 REM  ****************************************************************
280 REM         fin de la creacion de la tabla de respuestas
281 REM  ****************************************************************
282 REM
283 REM
284 REM  ****************************************************************
285 REM  *********** RUTINA DE CREACION DE VENTANAS  ********************
286 REM  ****************************************************************
287 LET F1=2
288 LET F2=15
289 LET C1=15
290 LET C2=65
291 CLS
292 LOCATE F1,C1
293 PRINT CHR$(201);
294 FOR V=C1+1 TO C2-1
295   PRINT CHR$(205);
296 NEXT V
297 PRINT CHR$(187)
298 FOR V=F1+1 TO F2-1
299   LOCATE V,C1
300   PRINT CHR$(186)
301   LOCATE V,C2
302   PRINT CHR$(186)
303 NEXT V
304 LOCATE F2,C1
305 PRINT CHR$(200);
```

```
306 FOR V=C1+1 TO C2-1
307   PRINT CHR$(205);
308 NEXT V
309 PRINT CHR$(188)
310 RETURN
311 REM
312 REM
313 REM  **************************************************************
314 REM          fin de la rutina de creacion de ventanas
315 REM  **************************************************************
```

TECNICAS DE ANALISIS

DISEÑO DE SISTEMAS (II)

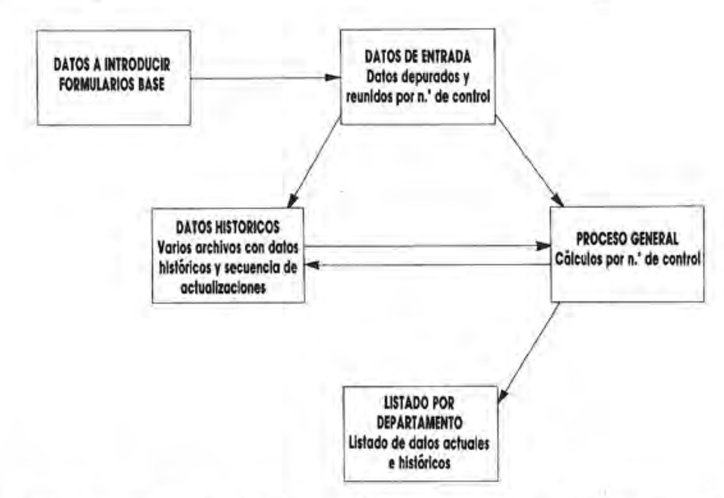


Organigrama general



EL objeto del organigrama general es plasmar de un modo gráfico las relaciones existentes entre los distintos elementos del sistema. Es importante que pueda ser observdo el conjunto de la aplicación simultáneamente (o en dos partes) por lo que debe incluirse en un solo gráfico (o en dos hojas a lo sumo) el organigrama completo del sistema que se está describiendo.

En este organigrama general se suele poner el acento en la presentación de los diferentes «grupos de datos» que intervienen en el proceso (es decir, archivos de entrada y/o salida e informes a obtener) y las relaciones de estos «grupos de datos» entre sí y con los procesos a realizar sobre ellos. De este modo, en el organigrama general suele aparecer cada proceso (cada programa) o cada grupo de funciones a realizar como una unidad, aunque comporte varias actividades o manipulaciones de los datos. Normalmente cada uno de estos «bloques» formados por un proceso y los archivos a él vinculados suele venir presentado en otro organigrama de detalle y constituye una unidad de tratamiento descrita independientemente.





Organigrama general representativo de las relaciones entre los datos y los procesos.

Más adelante estudiaremos en detalle cómo se realizan estos organigramas: simbología, convenciones a respetar, tipos, etc.



Organigramas de detalle

Cada una de las partes del sistema que se diseña (normalmente uno o dos

bloques del organigrama general) se suelen presentar más concretamente en un organigrama de detalle. El aspecto de estos organigramas es semejante al ya indicado para los organigramas generales, aunque incluyen especificaciones más concretas de los archivos e informes a obtener así como de los procesos a realizar con los datos.

En la figura 2 se presenta un posible organigrama de detalle del «proceso general de cálculo por número de control» para el sistema de la figura 1.

MOVIMIENTO DE ENTRADA
CLASIFICACION de los datos de entrada por n.° de control
FICHERO TEMPORAL
RECUPERACION DE LA POSICION histórica de cada n.° control
HISTORICO GENERAL
CALCULO de la nueva posición obtenida a partir de los nuevos datos de entrada
GRABACION de datos: en el archivo histórico de movimientos y en el histórico general
HISTORICO DE MOVIMIENTO
PARAMETROS PARA LOS INFORMES
PREPARACION de los informes a obtener
IMPRESION de informes
MOVIMIENTOS por fechas
MOVIMIENTOS por n.° de control
SITUACIÓN de los n.° de control seleccionados
DATOS GENERALES del proceso, estadísticos, etc.

Organigrama de detalle.

Es usual que estos organigramas de detalle sean reproducción de los que se incluyen en los dossieres de los diferentes programas.



Diseño de documentos de salida

Aunque en la documentación específica de cada programa se incluyan todos los formatos de los documentos que se obtienen en cada programa, se adjunta en el diseño global del sistema una información general de los documentos a obtener, con información de los datos que en ellos se van a incluir.

Se adjunta (fig. 3) un diseño de un posible formulario en el que se pueden incluir los datos de descripción del impreso.

IMPRESO DE ....

Impreso n.º ....
Utilidad ....

Formato:
Colum/línea: ....
Líneas/pág.: ....
Formato de control de carro: ....
Numeración de páginas: ....

Secuencia de impresión: ....

Niveles de impresión: ....

Copias: ....

Instrucciones de distribución: ....

Preparado por ....
Supervisado por ....
Aprobado por ....
Fecha ....

DESCRIPCION DE CAMPOS						
Nombre del campo	Formato de edición	Tamaño	Posiciones de impresión	Origen	Nivel	Descripción



Formulario de la descripción de impreso.

En él se suele poner, en la parte superior, el nombre del impreso para su fácil identificación por personas no conocedoras de la aplicación, aunque luego se le asigne un número codificado para su exacta referencia. El apartado «utilidad» se utiliza para describir (brevemente) el objeto del impreso y el tipo de información que en él se va a dar. Además, hay que incluir una descripción general del «aspecto» del impreso (número de columnas, línea, número de líneas por página, etc.). Es normal describir aparte el formato de saltos, cabeceras, subtotales, etcétera, en un documento específico cuyo nombre y/o número se debe dar aquí.

En la «Secuencia de impresión» se suele indicar el orden y secuencia en que se procesan los registros de datos de entrada para elaborar los informes de salida. Por otro lado, conviene describir (apartado «Niveles de impresión») bajo qué condiciones se producirá la rotura de control necesaria para la presentación de los datos.

La parte más importante del documento es la «Descripción de los campos»: en esta sección se describe el aspecto concreto de los campos que se incluyen en el informe o listado.

Nombre del campo. Se refiere al que se da a ese dato; normalmente suele ser el nombre que aparece en la cabecera de esa columna del impreso.

Formato de edición. Si el campo es alfabético (ajuste a la derecha) o numérico (ajuste a la izquierda) y, en este último caso, número de posiciones decimales, si se suprimen los ceros de la izquierda, cómo se pone el signo, etc.

Tamaño del campo.

Posiciones de impresión sobre las que se va a imprimir el campo correspondiente.

Origen de los datos con los que se elabora el contenido de este campo o, sencillamente, campo de un registro cuyo contenido se imprime aquí.

Nivel es el correspondiente «nivel de impresión» (de entre los descritos en la cabecera de este formulario) que corresponde al campo de que se trata.

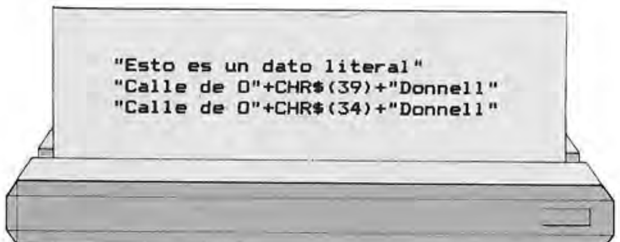
Descripción somera del contenido del campo, cuando de los datos anteriores (y especialmente del «nombre del campo») no se deduce claramente. Se pueden incluir aquí comentarios generales sobre el contenido de los diferentes campos o sobre su impresión.

TECNICAS DE PROGRAMACION

TIPOS DE DATOS (continuación)



VEAMOS algunos ejemplos de datos literales en el lenguaje BASIC:



Se observará que dentro de un dato literal (entre las dobles comillas) pueden incluirse letras mayúsculas y minúsculas mezcladas, en aquellos ordenadores que acepten ambos juegos de letras. Fuera de las comillas, los intérpretes de todas las letras minúsculas a mayúsculas. En los ejemplos puede verse también cómo se pueden mezclar en el mismo dato literal expresiones entre comillas y caracteres definidos mediante la función CHR$. Así, el segundo ejemplo se convertirá en «Calle de O'Donnell», mientras que el tercero utiliza la doble comilla en lugar de la comilla simple, por lo que viene a representar «Calle de O''Donnell».

En BASIC es posible asignarle a una variable un dato literal con una condición: que el hombre de la variable termine en el carácter $. Por tanto, serán variables literales válidas las siguientes:



Obsérvese que también pueden definirse agregados de datos literales, como la serie o vector B$, que tiene tres elementos, cada uno de los cuales puede tomar como valor una cadena de caracteres que normalmente está limitada a un máximo total de 255, incluyendo letras, cifras, símbolos y el espacio en blanco. También pueden definirse tablas de caracteres, como la variable TABLA$, que tiene cuatro filas y cuatro columnas. Cada uno de sus elementos (por ejemplo, TABLA$(2,3), el situado en la intersección de la segunda fila y la tercera columna), puede tomar como valor una cadena de caracteres cualquiera, dentro de los límites permitidos por el intérprete o compilador correspondiente.

Nota: En todos los intérpretes de BASIC existe un límite al número de caracteres que puede tener el nombre de una variable. En algunos este límite es muy pequeño. Es posible, por tanto, que algunos de los nombres dados como ejemplo en este capítulo y el anterior no sean válidos en un intérprete de BASIC determinado.

Al igual que vimos en el capítulo anterior al hablar de los distintos tipos de da-

tos numéricos, algunos intérpretes de BASIC permiten definir que todas las variables que comiencen por ciertas letras y no terminen en uno de los caracteres especiales de selección de tipo pertenezcan automáticamente al tipo literal. Esto se consigue con la instrucción DEFSTR («define string»), que funciona igual que DEFDBL y DEFINT, ya explicadas.

En el lenguaje PASCAL los datos literales se representan igual que en BASIC, con la diferencia de que se utiliza la comilla simple (') en lugar de la doble ("). Una comilla simple puede aparecer dentro de una cadena de caracteres, pero para no confundirla con la que señala el final de la cadena, deberá escribirse duplicada. Por tanto, las tres cadenas de caracteres que vimos como ejemplo en el caso del BASIC se escribirían en PASCAL así:



En algunos compiladores de PASCAL es posible obtener cualquier carácter ASCII de una forma parecida a la de la función CHR$ de BASIC. Sin embargo, en lugar de utilizar una función se suele conseguir esto mediante un símbolo especial seguido por el número ASCII del carácter que se desea obtener.

Como en PASCAL hay que declarar todas las variables, también las que contengan caracteres deberán aparecer declaradas al principio del programa. Veamos algún ejemplo:

```
program EJEMPLO;
(* Declaración de variables literales *)
var
  a: char;
     (* a contendrá un sólo carácter *)
  b: array[1..255] of char;
     (* b contiene 255 caracteres *)
  tabla: array[1..4,1..4] of char;
     (* tabla de 4 filas y 4 columnas *)
```

Hay una diferencia fundamental entre los datos literales de BASIC y los de PASCAL. En el primero de estos lenguajes, la unidad fundamental es la cadena de caracteres, cuya longitud puede variar entre ciertos límites (normalmente cero y 255). En PASCAL, sin embargo, la unidad es el carácter individual, con el que también se pueden formar vectores y matrices. Por eso, aunque en BASIC el valor de TABLA$(2,3) es una cadena de caracteres completa (por ejemplo, "MARTES") en PASCAL el valor de TABLA(2,3) será un solo carácter.

No obstante, en algunos compiladores de PASCAL se introduce un segundo tipo de datos literales además de «char», que se llama tipo «string», y que corresponde al tipo literal de BASIC. Aunque éste no es un tipo estándar del PASCAL, por lo que no es obligado que todos los compiladores lo entiendan, está bastante extendido. Veamos algunos ejemplos de su uso:

```
program EJEMPLO2;
(* Declaración de variables string *)
var
  a, nombre: string;
  b: array[1..3] of string;
  tabla: array[1..4,1..4] of string;
```

Los ejemplos indicados son equivalentes a los que vimos antes al tratar los datos literales en BASIC, aunque el modo de realizar la declaración puede variar ligeramente de compilador en compilador.

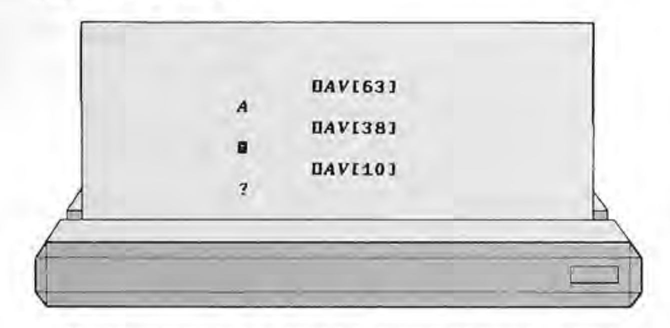
En el lenguaje APL los datos literales se introducen entre comillas simples, como en PASCAL. También es preciso duplicar las comillas que han de formar parte de las cadenas de caracteres. Veamos cómo se forman en este lenguaje las tres cadenas de caracteres de nuestros ejemplos anteriores:

```
      'Esto es un dato literal'
Esto es un dato literal
      'Calle de O''Donnell'
Calle de O'Donnell
      'Calle de O"Donnell'
Calle de O"Donnell
```

Como se ve, es totalmente equivalente al caso del PASCAL. También pueden

obtenerse caracteres aislados indexando una variable del sistema, cuyo nombre se forma mediante un cuadrado seguido de las letras A y V, siglas de «Atomic Vector», o vector atómico, pues es así como se llama en APL el conjunto de todos los caracteres posibles. El índice que hay que dar para obtener cada carácter depende, en general, del intérprete de APL, pues algunos intérpretes no utilizan el sistema ASCII de representación de caracteres. En los ejemplos dados a continuación utilizaremos el sistema de representación interna del intérprete APL de IBM para el IBM Personal Computer.

En primer lugar, para obtener un solo carácter basta con indexar la variable «cuadrado-AV» con el número de orden que ocupa ese carácter en la lista de caracteres APL.

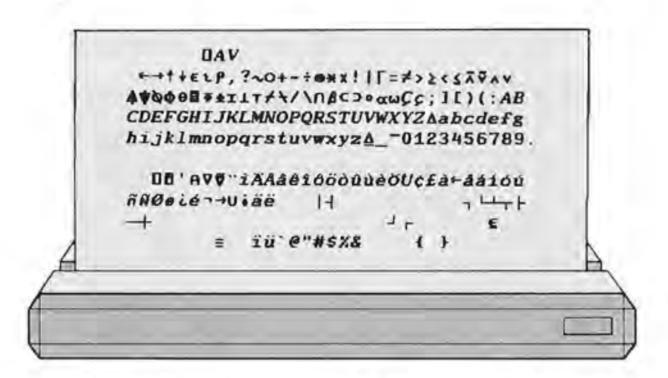


Para obtener varios caracteres consecutivos puede indexarse a la vez el conjunto de todos los caracteres («cuadrado AV») por los números de orden correspondientes:

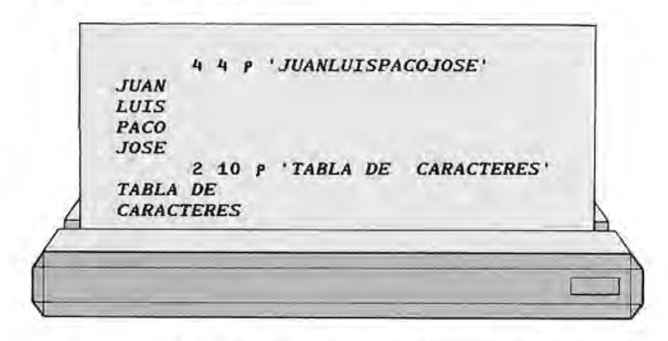


Obsérvese que los índices pueden darse en forma de lista de datos (como en el caso 63, 38 10) o de una expresión APL cualquiera. Como se recordará, la letra griega «iota» colocada delante de un número genera todos los números comprendidos entre 1 y el número dado. En nuestro caso, los números de 1 a 26. Si a esa lista de números le sumamos 62, obtendremos una nueva lista de números comprendidos entre 63 y 88, que son precisamente los índices de las 26 letras mayúsculas del alfabeto en la representación interna APL que estamos considerando.

Por último, podemos también obtener la lista de todos los caracteres posibles (y, por tanto, deducir su orden) invocando la variable «cuadrado-AV» sin índice alguno:



Los datos literales APL pueden formar también series y tablas (vectores y matrices), igual que ocurre con los datos numéricos. Las primeras se obtienen, como ya hemos visto, escribiendo una cadena de caracteres entre comillas o indexando la variable del sistema «cuadrado-AV» por una serie de índices: Las tablas o matrices literales se construyen como vimos en el capítulo cuarto, utilizando la letra griega «rho», a cuya izquierda se coloca el número de filas y columnas que va a tener la tabla y a su derecha los valores correspondientes ordenados por filas, en forma de serie de caracteres. Veamos algunos ejemplos:



Como ya hemos dicho varias veces, en APL no es preciso declarar las variables ni existe regla alguna sobre qué tipo de datos puede contener cada variable. Cualquier nombre es compatible con cualquier tipo de datos. En general, una variable será numérica o literal según los valores concretos que se le asignen. Si es numérica, será entera o real, dependien-

do de los valores que tome. Igualmente ocurre con su estructura: podrá ser un escalar (un solo valor), un vector (una serie de valores) o una matriz (una tabla de filas y columnas) dependiendo del valor que se le haya asignado, que puede cambiar a lo largo de un programa. Por tanto, la misma variable, que en cierta instrucción podía ser una matriz de caracteres, en otra podrá convertirse en un vector de números. Esta práctica no es aconsejable, pues tiende a hacer los programas más difíciles de leer por otras personas, o incluso por el mismo programador, si hace cierto tiempo que los escribió.



Otros tipos y estructuras de datos

Además de los tipos y estructuras de datos mencionados en las páginas anteriores, existen otros muchos. Entre las estructuras, podemos mencionar las siguientes:

1. *Estructuras rectangulares de orden superior.* Son equivalente a los vectores y matrices, pero tienen más de dos dimensiones. Su nombre técnico, en matemáticas, es «tensores». En Informática se suelen denominar «agregados multidimensionales» (*arrays*, en inglés). Se crean o declaran con el mismo tipo de instrucciones utilizadas para declarar o crear matrices, pero dando el número de dimensiones adecuado para cada caso. Así, en BASIC se utilizará la instrucción DIM, en PASCAL la declaración de tipo «array» y en APL la letra griega «rho».
2. *Estructuras arborescentes.* Listas y árboles. Existen diversas técnicas para construirlas, en las que no vamos a entrar aquí en este momento.

3. *Estructuras gráficas.*

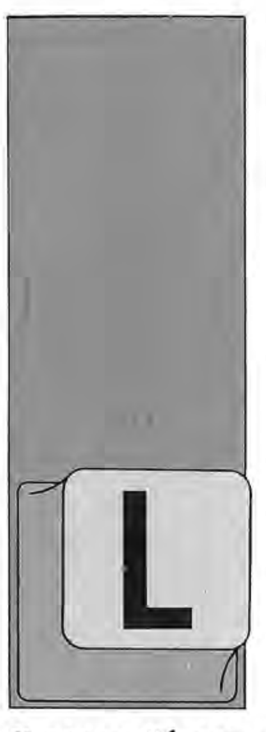
Mencionemos también algunos tipos de datos:

1. *Datos lógicos.* Muy utilizados. Hablaremos de ellos con detalle más adelante.
2. *Nombres o identificadores.* Este tipo de datos no existe en BASIC o en APL, donde, en general, se sustituyen por los tipos literales. Son frecuentes en PASCAL.
3. *Punteros.* Son muy utilizados en los lenguajes próximos a la máquina, como el lenguaje simbólico o el C.

Algunos lenguajes, como el PASCAL, permiten incluso definir tipos nuevos, así como subtipos de los tipos fundamentales o de los definidos por el programador.

APLICACIONES

TRATAMIENTO DE TEXTOS
EJEMPLOS CON EL PROGRAMA WORDSTAR



LA mejor manera de aprender a utilizar un programa de aplicaciones es sentarse frente al ordenador y realizar ejemplos que nos permitan ir conociendo su utilización. Si procediéramos memorizando una lista de comandos, posiblemente no llegaríamos a conocer bien el programa; sin embargo, al trabajar en casos prácticos veremos cómo es innecesario memorizar nada, simplemente nos acostumbraremos a hacer que nuestro programa sea una herramienta de trabajo.

En este fascículo presentamos ejemplos con los que se puede "jugar" para familiarizarse con el programa; lo iremos describiendo paso a paso para facilitar la labor.

Enviemos una circular

El primer paso que vamos a dar es crear la carta que deseamos enviar. Para ello cargamos el WordStar (WS) en nuestro ordenador y vemos aparecer el "menú sin archivo"; entre las opciones que presenta están la de abrir un documento y la de cambiar la unidad de disco activa. Aconsejamos antes de empezar cambiar a la unidad B>, si se dispone de ella; de esta forma no existe la posibilidad de perder cualquier información existente en el disco maestro del WordStar. Para ello pulsamos L y a la pregunta de cuál queremos que sea la nueva unidad de disco respondemos b: (intro).

Ya estamos trabajando en la unidad B y podemos abrir nuestro documento:

— Pulsamos D.

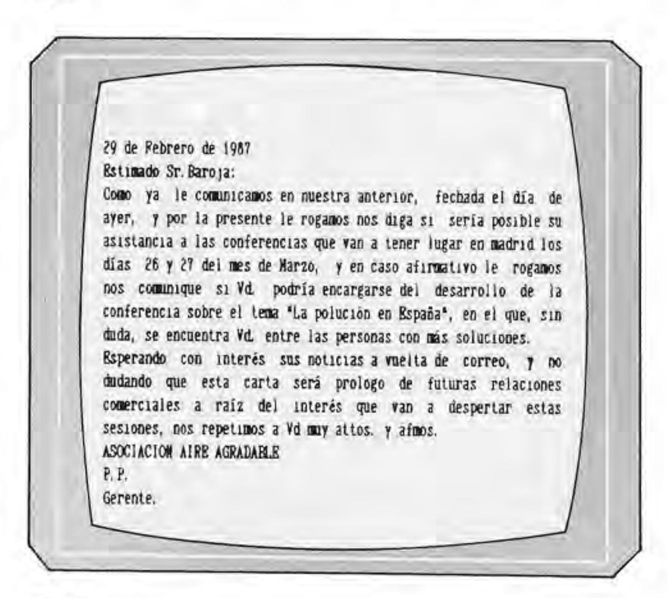
— WordStar pregunta por el nombre del documento que queremos abrir.

— Respondemos: CIRCULAR (Intro).

— WordStar nos comunica que se trata de un documento nuevo.

El siguiente paso consiste en escribir el texto que queremos que aparezca; como se trata de una carta que mandaremos a varias personas, vamos a poner el nombre y los datos correspondientes a la primera de ellas.

La forma de escribir en un procesador de textos es muy sencilla; sólo tendremos que preocuparnos de pulsar (Intro) después de un punto y aparte; mientras tanto, escribiremos continuamente y sin preocuparnos de que la línea se termina o de cómo se distribuyen las palabras; esta tarea corresponde a un paso posterior.



29 de Febrero de 1987

Estimado Sr. Baroja:

Como ya le comunicamos en nuestra anterior, fechada el día de ayer, y por la presente le rogamos nos diga si sería posible su asistancia a las conferencias que van a tener lugar en madrid los días 26 y 27 del mes de Marzo, y en caso afirmativo le rogamos nos comunique si Vd. podría encargarse del desarrollo de la conferencia sobre el tema "La polución en España", en el que, sin duda, se encuentra Vd. entre las personas con más soluciones.

Esperando con interés sus noticias a vuelta de correo, y no dudando que esta carta será prologo de futuras relaciones comerciales a raíz del interés que van a despertar estas sesiones, nos repetimos a Vd muy attos. y afmos.

ASOCIACION AIRE AGRADABLE

P.P.

Gerente.



Primer aspecto de la carta. Sólo se ha pulsado (Intro) después de los puntos y aparte.

Cuando la carta está terminada podemos pasar a reformar el texto para que quede a nuestro gusto. Para ello nos situamos en modo INSERTAR, que puede variarse con (control)V o simplemente presionando la tecla de inserción.

— Insertando espacios en blanco con el tabulador se sitúa la fecha en su sitio.

— Para el bloque central de la carta podemos utilizar dos sistemas:

a) Con F1 (F2 en la versión en inglés) variamos el margen izquierdo de acuerdo con los tabuladores; cuando se encuentre el margen en el lugar que deseamos, presionaremos (Control)B y el texto se reforma hasta el siguiente punto y aparte. Este proceso se repetirá hasta el final del texto. Si WordStar encontrara una palabra que quisiera partir al terminar una línea, nos preguntaría si el lugar es apropiado, de forma que nunca se partirá una sílaba por la mitad.

B:CIRCULAR PAG. 1 LIN. 17 COL 62 INSERTAR SI

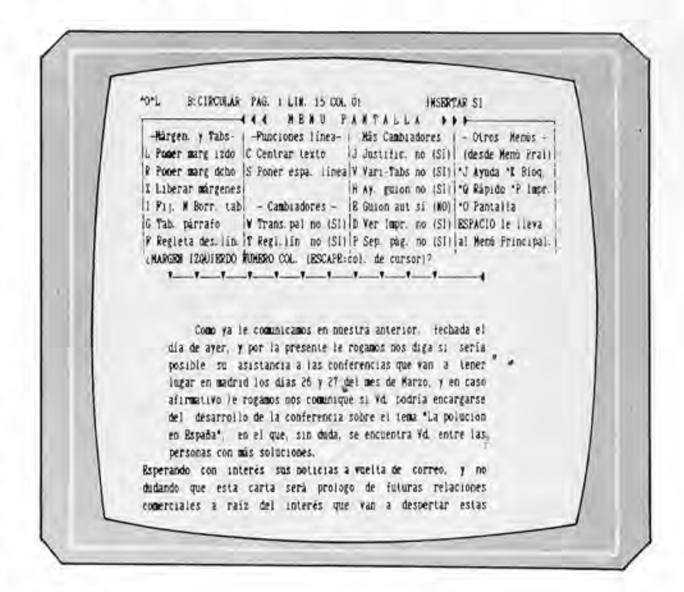
PARA PONER GUION, PULSE -. Antes de pulsar -, se puede
mover el cursor: ^S:cursor izda., ^D:cursor dcha.
Si no desea guion, pulse ^B.

Como ya le comunicamos en nuestra anterior, fechada el
día de ayer, y por la presente le rogamos nos diga si
sería posible su asistancia a las conferencias que van a
tener lugar en madrid los días 26 y 27 del mes de Marzo, y
en caso afirmativo le rogamos nos comunique si Vd. podría
encargarse del desarrollo de la conferencia sobre el tema
"La polucion en España", en el que, sin duda, se encuentra
Vd. entre las personas con más soluciones.

Esperando con interés sus noticias a vuelta de correo, y
no dudando que esta carta será prologo de futuras relaciones
comerciales a raíz del interés que van a despertar estas
sesiones, nos repetimos a Vd muy attos. y afmos.

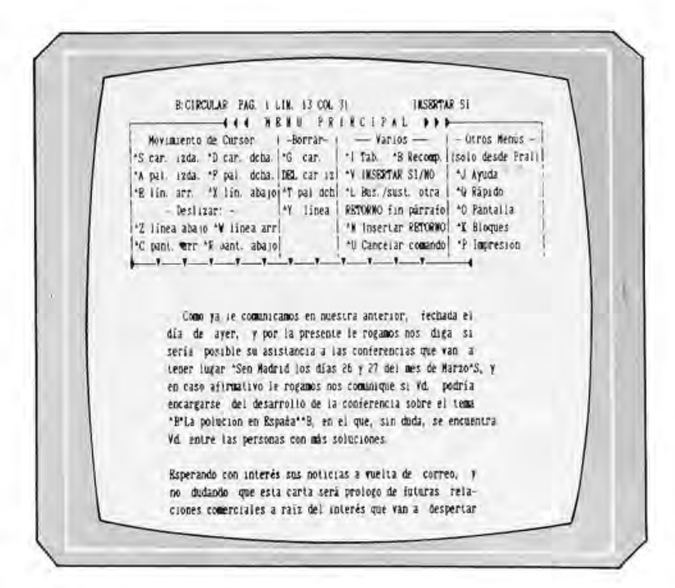
Reformando el texto. WordStar pregunta si puede partir la sílaba.

b) Variamos el margen izquierdo situando el cursor en la columna que deseamos poner dicho margen y pulsamos (Control)OL y (Escape) (o F2); observará cómo varía la posición del margen. Estando éste ya fijo, se puede proceder a reformar el texto con (Control)B y si queremos insertar una línea simplemente pulsamos (Intro).

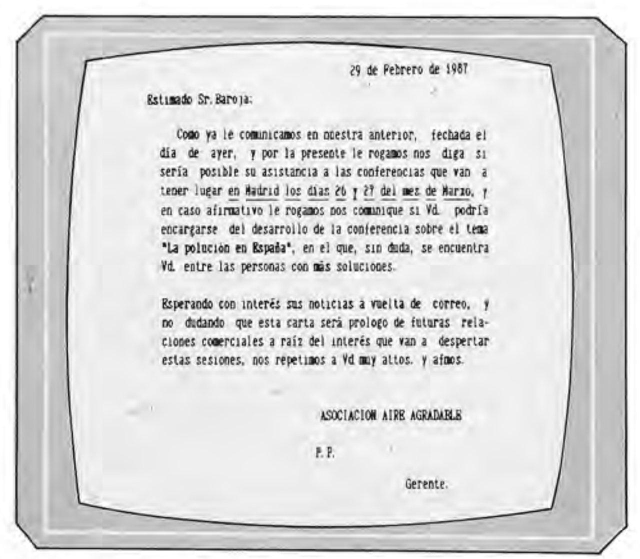


Reformando el texto. Con cambio del margen.

Podemos añadir para dar vistosidad a la carta algunos atributos como subrayar una parte del texto ((^)S) o poner en negrilla o doble pasada ((^)B). Por otra parte, antes de la impresión releemos el texto para corregir posibles errores (ejemplo: donde pone madrid hay que poner Madrid).



Algunas características introducidas en el texto.



29 de Febrero de 1987

Estimado Sr.Baroja:

Como ya le comunicamos en nuestra anterior, fechada el día de ayer, y por la presente le rogamos nos diga si sería posible su asistancia a las conferencias que van a tener lugar en Madrid los días 26 y 27 del mes de Marzo, y en caso afirmativo le rogamos nos comunique si Vd. podría encargarse del desarrollo de la conferencia sobre el tema "La polución en España", en el que, sin duda, se encuentra Vd. entre las personas con más soluciones.

Esperando con interés sus noticias a vuelta de correo, y no dudando que esta carta será prologo de futuras relaciones comerciales a raíz del interés que van a despertar estas sesiones, nos repetimos a Vd muy attos. y afmos.

ASOCIACION AIRE AGRADABLE

P.P.

Gerente.



Aspecto que tendra la carta. Se consigue verlo en pantalla con (Control)OD.

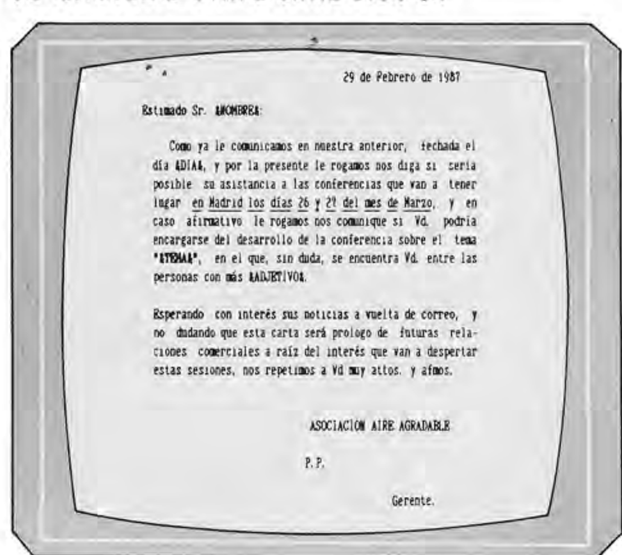


Utilización del MailMerge

MailMerge fue diseñado para permitir a los usuarios automatizar sus listas de correos, así como realizar una composi ción de ficheros.

Continuando con nuestro ejemplo, vamos a aplicar al mismo el programa MailMerge.

Para empezar, se transforma el documento que estábamos escribiendo. Se sustituyen las palabras que deberán ser variadas para cada destinatario por un identificador o nombre de variable que se encierra entre símbolos &.



29 de Febrero de 1987

Estimado Sr. &NOMBRE&:

Como ya le comunicamos en nuestra anterior, fechada el día &DIA&, y por la presente le rogamos nos diga si sería posible su asistancia a las conferencias que van a tener lugar en Madrid los días 26 y 27 del mes de Marzo, y en caso afirmativo le rogamos nos comunique si Vd. podría encargarse del desarrollo de la conferencia sobre el tema "&TEMA&", en el que, sin duda, se encuentra Vd. entre las personas con más &ADJETIVO&.

Esperando con interés sus noticias a vuelta de correo, y no dudando que esta carta será prologo de futuras relaciones comerciales a raíz del interés que van a despertar estas sesiones, nos repetimos a Vd muy attos. y afmos.

ASOCIACION AIRE AGRADABLE

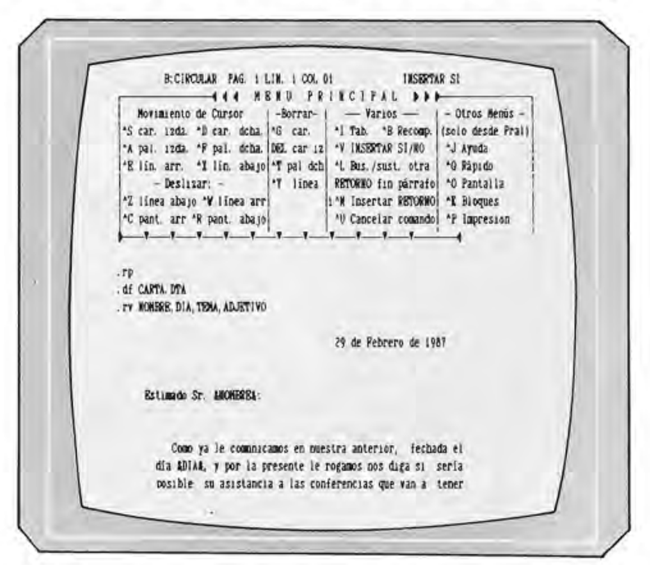
P.P.

Gerente.



Nuevo aspecto de la circular para la utilización de MailMerge.

Se introducen unas instrucciones de punto propias de MailMerge (WordStar visualiza una M en la columna de señales).



```
B:CIRCULAR  PAG. 1 LIN. 1 COL 01            INSERTAR SI
               ◄◄◄ M E N U   P R I N C I P A L ►►►
 Movimiento de Cursor     | -Borrar-  |   — Varios —         | - Otros Menús -
 ^S car. izda. ^D car. dcha.| ^G car.  | ^I Tab.  ^B Recomp.  | (solo desde Pral)
 ^A pal. izda. ^F pal. dcha.| DEL car iz| ^V INSERTAR SI/NO   | ^J Ayuda
 ^E lín. arr.  ^X lín. abajo| ^T pal dch| ^L Bus./sust. otra  | ^Q Rápido
     - Deslizar: -          | ^Y línea  | RETORNO fin párrafo | ^O Pantalla
 ^Z línea abajo ^W línea arr|           | ^N Insertar RETORNO | ^K Bloques
 ^C pant. arr ^R pant. abajo|           | ^U Cancelar comando | ^P Impresion

.rp
.df CARTA.DTA
.rv NOMBRE,DIA,TEMA,ADJETIVO

                                   29 de Febrero de 1987

     Estimado Sr. &NOMBRE&:

        Como ya le comunicamos en nuestra anterior, fechada el
     día &DIA&, y por la presente le rogamos nos diga si sería
     posible su asistancia a las conferencias que van a tener
```



Instrucciones de punto para el documento al que vamos a aplicar MailMerge.

El significado de las instrucciones de punto es el siguiente:

1) .rp indica a MailMerge que deberá repetir la impresión.

2) .df indica en qué fichero se van a encontrar los datos.

3) .rv indica las variables que se deben leer.

4) .pa indica que la página ha terminado, se sitúa al final.

Una vez introducidas estas modificaciones podemos salvar el documento CIRCULAR utilizando el comando (Control)KD que nos devuelve al menú sin archivo.

Queda crear el fichero de datos con toda la información necesaria para cada una de las cartas. Este es un fichero sencillo que se crea usando el modo NON-DOCUMENT de WordStar y que consiste en una serie de registros. Cada registro es una línea terminada al pulsar (Intro). Cada campo del registro se encuentra separado de los adyacentes por comas.

Creamos en nuestro caso el fichero CARTAS.DTA, para ello pulsamos desde el menú sin archivo N que nos permite abrirlo pidiéndonos el nombre del mismo modo que si se tratara de un documento normal.

Introducimos los registros, sin olvidar que es necesario dar a (Intro) cada vez

que termina uno. Salvamos el fichero con (Control)KD para acceder nuevamente al menú principal.



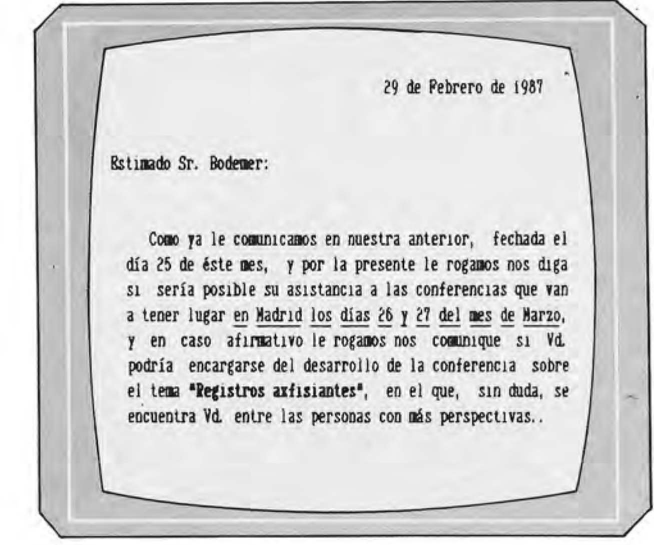
Baroja, de ayer, La polucion en España, soluciones
D. Mariano Bearlintz, 26 de éste mes, Distorsion del ruido en la atmosfera, iniciativas
Cantero, de ayer, Delegaciones de aire, conocimientos
Bodemer, 25 de éste mes, Registros axfisiantes, perspectivas.



Fichero de datos para la circular.

Ya podemos ejecutar MailMerge; nos pide el nombre del fichero con el que queremos que actúe, si presionamos (Intro), sale en pantalla una lista de los posibles, introducimos el nombre del nuestro, CIRCULAR, y respondemos a las preguntas necesarias para la impresión. Ya tenemos la circular preparada para ser enviada a los diferentes destinatarios.

Continúa con otros ejemplos el estudio del WordStar; como puedes comprobar te va a resultar de gran ayuda en tu traba jo y en tu correspondencia. Si profundizas más verás cómo existen otras ventajas y trucos que cada vez te serán más útiles.



29 de Febrero de 1987

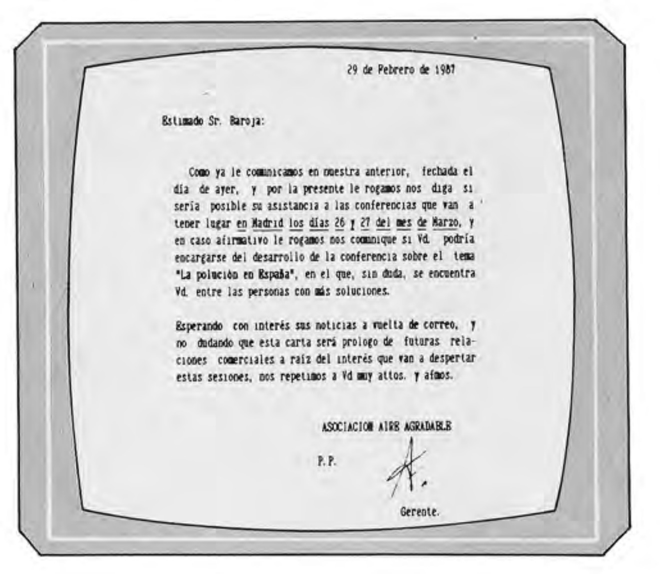
Estimado Sr. Bodemer:

Como ya le comunicamos en nuestra anterior, fechada el día 25 de éste mes, y por la presente le rogamos nos diga si sería posible su asistancia a las conferencias que van a tener lugar en Madrid los días 26 y 27 del mes de Marzo, y en caso afirmativo le rogamos nos comunique si Vd. podría encargarse del desarrollo de la conferencia sobre el tema **"Registros axfisiantes"**, en el que, sin duda, se encuentra Vd. entre las personas con más perspectivas..

29 de Febrero de 1987

Estimado Sr. D. Mariano Bearlintz:

Como ya le comunicamos en nuestra anterior, fechada el día 26 de éste mes, y por la presente le rogamos nos diga si sería posible su asistancia a las conferencias que van a tener lugar en Madrid los días 26 y 27 del mes de Marzo, y en caso afirmativo le rogamos nos comunique si Vd. podría encargarse del desarrollo de la conferencia sobre el tema **"Distorsión del ruido en la atmósfera"**, en el que, sin duda, se encuentra Vd. entre las personas con más iniciativas.



29 de Febrero de 1987

Estimado Sr. Baroja:

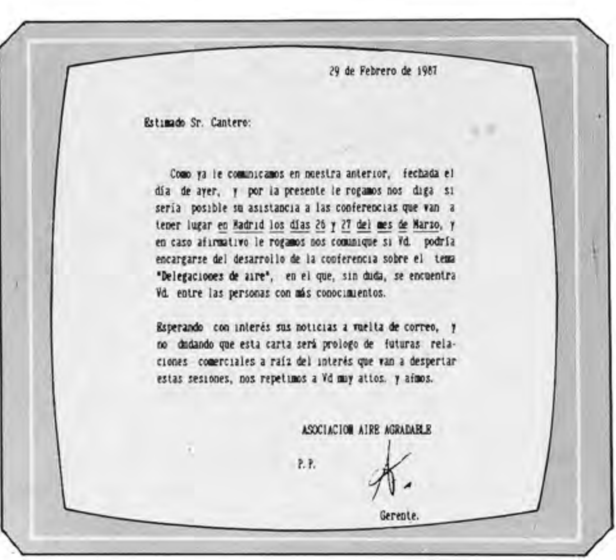
Como ya le comunicamos en nuestra anterior, fechada el día de ayer, y por la presente le rogamos nos diga si sería posible su asistancia a las conferencias que van a tener lugar en Madrid los días 26 y 27 del mes de Marzo, y en caso afirmativo le rogamos nos comunique si Vd. podría encargarse del desarrollo de la conferencia sobre el tema **"La polución en España"**, en el que, sin duda, se encuentra Vd. entre las personas con más soluciones.

Esperando con interés sus noticias a vuelta de correo, y no dudando que esta carta será prologo de futuras relaciones comerciales a raíz del interés que van a despertar estas sesiones, nos repetimos a Vd muy attos. y afmos.

ASOCIACION AIRE AGRADABLE

P.P.

Gerente.



29 de Febrero de 1987

Estimado Sr. Cantero:

Como ya le comunicamos en nuestra anterior, fechada el día de ayer, y por la presente le rogamos nos diga si sería posible su asistancia a las conferencias que van a tener lugar en Madrid los días 26 y 27 del mes de Marzo, y en caso afirmativo le rogamos nos comunique si Vd. podría encargarse del desarrollo de la conferencia sobre el tema **"Delegaciones de aire"**, en el que, sin duda, se encuentra Vd. entre las personas con más conocimientos.

Esperando con interés sus noticias a vuelta de correo, y no dudando que esta carta será prologo de futuras relaciones comerciales a raíz del interés que van a despertar estas sesiones, nos repetimos a Vd muy attos. y afmos.

ASOCIACION AIRE AGRADABLE

P.P.

Gerente.

Resultado final de la aplicación de MailMerge.



PASCAL

PROCEDIMIENTOS Y FUNCIONES

CUANDO escribimos el programa que dibujaba figuras, lo hicimos siguiendo más o menos la técnica denominada «de refinamiento gradual», es decir, descomponiendo primero el problema en sus partes principales y analizando luego cada una de estas partes por separado para descomponerlas a su vez en tareas más sencillas. Así, teníamos un primer esquema del programa como éste:

Repetir lo siguiente...
1. **Borrar la pantalla.**
2. **Preguntar qué tipo de figura se desea.**
3. **Preguntar de qué tamaño.**
4. **Dibujar la figura deseada.**
5. **Preguntar si se desea seguir.**

... hasta que no se desee seguir.

A su vez, «dibujar la figura deseada» era aproximadamente así»:

— Si se desea un cuadrado entonces:
Pintar un cuadrado.
pero si no, entonces:
— Si es una pirámide lo deseado:
Pintar una pirámide.
Y si tampoco es una pirámide:
Pintar un rombo.

El proceso de pintar una pirámide, por ejemplo, se podría descomponer aún más.

Como culminación de este proceso, hubo que juntar todos los pasos sencillos en que llegamos a descomponer el problema para construir el programa definitivo.

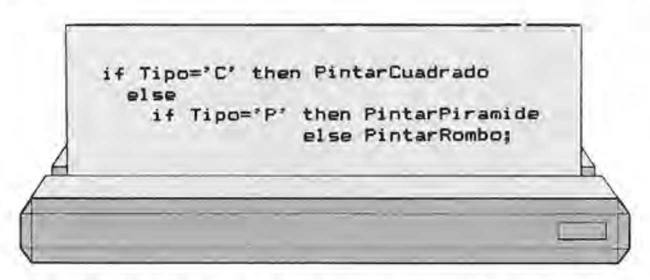
Si, teniendo un programa terminado, se descubren errores o se desea modificar alguna de sus partes integrantes, hay dos opciones: o bien se vuelve a repetir todo el proceso de descomposición y posterior integración, o bien se introducen los cambios directamente en el programa ya escrito; con programas tan cortos como los que llevamos hechos hasta el momento, cualquiera de los dos métodos es factible.

No es ésta la situación, sin embargo, cuando un programa es grande y complejo. Si optamos por la primera solución, desperdiciaremos una gran cantidad de trabajo ya hecho; por el contrario, si optamos por manipular el programa directamente, como ya están todas las partes integradas, habrá que delimitar claramente el área afectada antes de introducir los cambios y comprobar que éstos no afectan a las otras partes.

Esta tarea sería mucho más sencilla si el programa PASCAL reflejara los diferentes niveles de descomposición del problema. Así, la parte principal del programa se correspondería más o menos con el primer nivel de descomposición, indicando las diferentes fases del proceso. Luego, cada una de éstas estaría programada por separado, constituyendo lo que denominaríamos «subprogramas».

Al ejecutarse el programa, la parte principal iría utilizando a los diferentes subprogramas según fuera preciso. De esta manera, el punto 4 del esquema del

último programa lo podríamos escribir así:



PintarCuadrado, PintarPiramide y PintarRombo serían los nombres de los subprogramas donde ya sí estaría detallado lo que hay que hacer para pintar cada figura. En caso de querer hacer algún cambio en la forma de dibujar, por ejemplo, los rombos, no habría que tocar nada del programa principal; todos los cambios serían exclusivamente en el subprograma.

A su vez, el subprograma podría hacer referencia de manera análoga a otros subprogramas de nivel inferior que realizaran alguna de sus partes constitutivas.

Todo esto es posible en PASCAL mediante el uso de PROCEDIMIENTOS y FUNCIONES, que son conjuntos de instrucciones a lo que se les ha dado un nombre o identificador. Para ser utilizados, basta con escribir éste como una instrucción más, o sea, separado de las demás por punto y coma.

Cuando se ejecuta el programa, al llegar a un nombre de procedimiento o función se pasa a ejecutar su conjunto de instrucciones. Tras ellas se siguen ejecutando las que vengan a continuación del nombre.

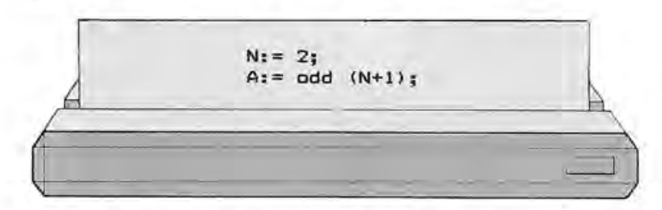
La única diferencia entre procedimientos y funciones es que éstas últimas además devuelven un dato (un número, un carácter...) para que lo utilice el programa principal.

En PASCAL existen procedimientos y funciones previamente programados. Un procedimiento que ya conocemos es PAGE (o CLRSCR). Cuando ponemos:



al ejecutarse PAGE, realmente se ejecuta un conjunto de instrucciones con ese nombre que sirven para borrar la pantalla. Tras ello, se ejecutaría la siguiente instrucción, WRITE.

También hemos utilizado la función ODD. Está formado por un conjunto de instrucciones que, al estar ya preparadas, no tienen que escribirse. A estas instrucciones se les suministra un dato de tipo INTEGER y, según que sea impar o no, devuelven el valor de tipo Boolean TRUE o FALSE, respectivamente. El dato a analizar se escribe a continuación del nombre de la función, entre paréntesis, y el dato Boolean devuelto pasa a «ocupar», por decirlo de alguna manera, el sitio del nombre de la función. Por ejemplo:



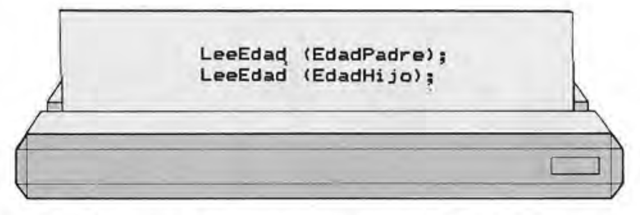
Al llegar al nombre ODD se pasa a ejecutar su conjunto de instrucciones, a las que se transfiere el número 3. Estas los analizan y devuelven TRUE, por lo que en ese momento esa instrucción equivaldría a A:= true.

Ejemplos de procedimientos a los que se transfieren datos para procesar son WRITE y READ, y funciones ya estudiadas son PRED, SUCC y ORD.

Otra ventaja de los procedimientos y funciones, casi tan importante como la de poder estructurar los programas, es la de permitir hacerlos más cortos.

En efecto, supongamos que hemos preparado una secuencia de instrucciones para, por ejemplo, pedir la edad de una persona controlando que se teclee una edad correcta. Si al escribir el programa resultase que hay que leer las edades de varias personas, no habría más remedio que repetir esas instrucciones para todos los casos necesarios.

En lugar de eso, podríamos agrupar esas instrucciones como un procedimiento de nombre, digamos, LeeEdad. Entonces bastaría con escribir LeeEdad en todos los puntos del programa necesarios; además, habría que indicarle cada vez al procedimiento en qué variable habría que guardar la edad leída. Por ejemplo:



```
LeeEdad (EdadPadre);
LeeEdad (EdadHijo);
```

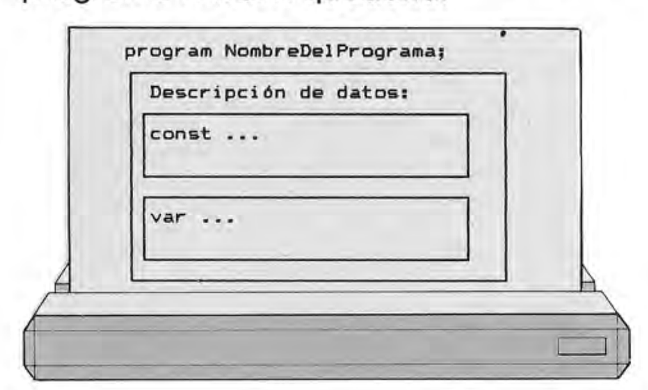
Como se ve, gracias a los procedimientos nos va a resultar posible escribir programas de una manera mucho más cómoda y, sobre todo, mucho más clara.



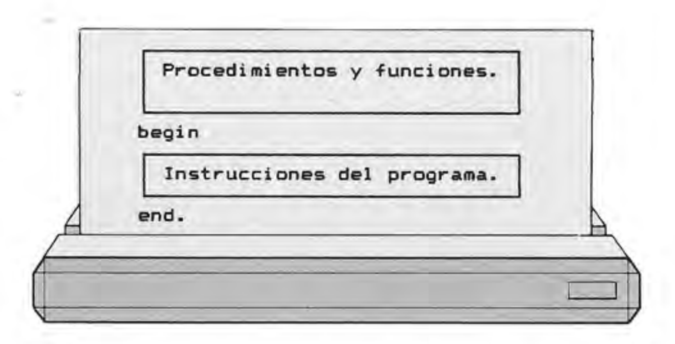
Cómo se escriben procedimientos

Los procedimientos se escriben después de la zona de descripción de datos del programa, justo antes de la palabra reservada BEGIN que marca el comienzo de la zona de instrucciones.

De esta manera, la estructura de un programa PASCAL quedaría:



```
program NombreDelPrograma;
  Descripción de datos:
    const ...
    var ...
```



```
  Procedimientos y funciones.
begin
  Instrucciones del programa.
end.
```

De las distintas partes del programa, sólo la cabecera y las palabras BEGIN y END con punto son obligatorias en todo caso. Las diferentes partes se separan entre sí por medio de punto y coma.

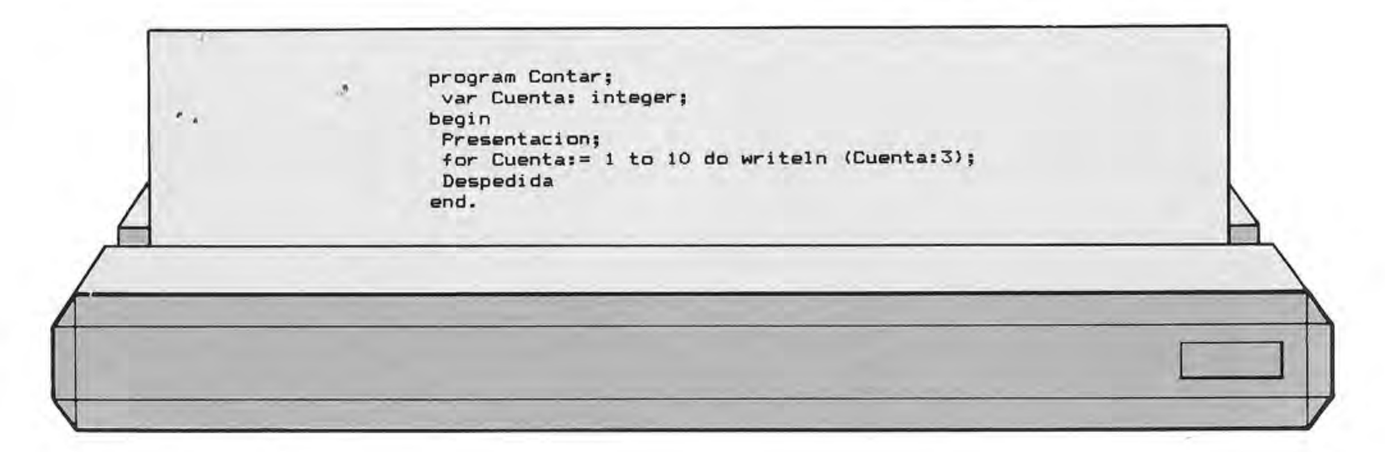
Supongamos que queremos escribir un programa que cuente del 1 al 1, tendría la siguiente estructura:

1. Presentar un mensaje en la pantalla indicando lo que hace.

2. Para CUENTA valiendo desde 1 hasta 10 hacer:
- **Presentar el valor de CUENTA en la pantalla.**
- **Pasa a la siguiente línea.**

3. Presentar un mensaje de despedida.

A modo de ejemplo, vamos a escribir los puntos 1 y 3 como procedimientos. Tendríamos un programa como:



```
program Contar;
 var Cuenta: integer;
begin
 Presentacion;
 for Cuenta:= 1 to 10 do writeln (Cuenta:3);
 Despedida
end.
```

Al intentar compilar el programa se producirían errores, pues al buscar el compilador las descripciones de los procedimientos no las encontraría.

La descripción de un procedimiento comienza por la palabra reservada PROCEDURE seguida de su nombre, que puede ser cualquier identificador válido. Esto es lo que se denomina CABECERA DEL PROCEDIMIENTO, que debe ir separada de lo siguiente por un punto y coma. Además, todo procedimiento consta de las palabras reservadas BEGIN y END que sirven para enmarcar sus instrucciones.

Las descripciones de los diferentes procedimientos y funciones se escriben unas detrás de otras, separándose la cabecera de uno de la palabra END del anterior mediante un punto y coma.

Por tanto, para que el compilador no produjera errores bastaría con poner:

```
program Contar;
 var Cuenta: integer;

procedure Presentacion;
 begin
 end;
procedure Despedida;
 begin
 end;

begin
 Presentacion;
 for Cuenta:= 1 to 10 do writeln (Cuenta:3);
 Despedida
end.
```

Al ejecutarse este programa, como la descripciones de los procedimientos no constan de ninguna instrucción, es como si hubiéramos puesto «no hacer nada», con lo que sólo saldrá la cuenta en la pantalla. Nuestra intención es que los procedimientos saquen unos mensajes y, por tanto, deben tener escritas las instrucciones necesarias para ello, con lo que el programa definitivo quedaría así:

```
program Contar;
 var Cuenta: integer;

procedure Presentacion;
 begin
  clrscr;   (* o PAGE  *)
  writeln ('Esto cuenta de 1 a 10.');
  writeln ('Y si no se lo creen, miren:')
 end;

procedure Despedida;
 begin
  writeln ('Yo casi nunca miento. Adiós.')
 end;

begin
 Presentacion;
 for Cuenta:= 1 to 10 do writeln (Cuenta:3);
 Despedida
end.
```

Por supuesto, las instrucciones utilizadas en los procedimientos van separadas entre sí por punto y coma, y pueden ser de cualquiera de los tipos estudiados.



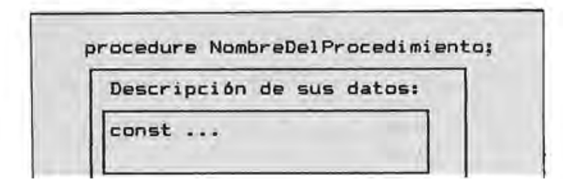
Estructura de un procedimiento

A los procedimientos y funciones se les conoce en general como subprogramas,

o sea, programas de orden menor que son utilizados por otros de orden superior.

Si recordamos cómo es la estructura de un programa, observaremos que la estructura de los procedimientos del ejemplo anterior es muy similar. En ambos casos existe una cabecera que comienza por las palabras reservadas PROCEDURE o PROGRAM, según el caso. Tras la cabecera viene el conjunto de instrucciones enmarcado por las palabras BEGIN y END, habiendo un punto a continuación de END en el caso de los programas para indicar el final absoluto y un punto y coma en los procedimientos para separarlos de lo que venga a continuación.

Pues bien, el paralelismo va mucho más lejos. Los procedimientos pueden tener su propia zona de definición de datos con variables para su uso exclusivo. Incluso pueden tener a su vez sus propios procedimientos y funciones. La estructura quedaría, por tanto, así:





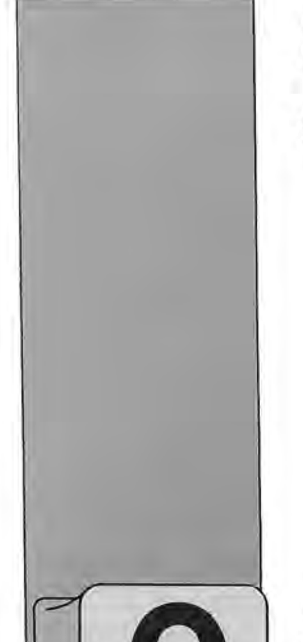
Es decir, son como programas PASCAL escritos dentro del programa principal.

Tanto las constantes como las variables, procedimientos y funciones definidos dentro de un procedimiento se denominan LOCALES de ese procedimiento y son para su uso exclusivo. Solamente pueden ser utilizados dentro de él (y de los propios procedimientos y funciones de éste).

Sin embargo, todas las constantes y variables del programa principal siguen pudiendo ser utilizadas por las instrucciones del procedimiento.

OTROS LENGUAJES

SISTEMAS OPERATIVOS: OASIS



OASIS es un sistema operativo multiusuario creado por la compañía Phase One en 1977, que está indicado, sobre todo, para mecanizar tareas de gestión de empresas y oficinas de pequeño y mediano tamaño.

La primera versión de OASIS (llamada OASIS 8, que es la que vamos a tratar), se diseñó para funcionar en equipos basados en el microprocesador Zilog Z-80, de 8 bits, y ya hay otra versión llamada THEOS OASIS 16, para equipos con microprocesador de 16 bits, tales como Intel 8088, 8086, 80286, Motorola 68000, etc.

OASIS 8	THEOS OASIS 16
— Funciona en sistemas basados en el microprocesador Zilog Z-80. — Monotarea. — Máximo de 16 usuarios.	— Funciona en equipos con microprocesadores de 16 bits, como Intel 8088/8086 y Motorola 68000. — Multitarea (hasta 256 trabajos a la vez). — Máximo de 32 usuarios.

Algunas diferencias entre OASIS y THEOS OASIS 16.

Estructura de OASIS

Como ya se ha dicho, OASIS 8 es un sistema operativo multiusuario que funciona en sistemas con una capacidad mínima de memoria RAM de 64 Kbytes. Soporta hasta un máximo de 16 usuarios, pero no soporta el trabajo en modo multitarea, es decir, no puede gestionar más de un proceso de forma concurrente.



Partes componentes del sistema operativo OASIS.

OASIS 8 se puede dividir en tres grandes partes:

1. *El núcleo del sistema.* Tiene una ocupación de memoria RAM entre 16 y 32 Kbytes. Sus funciones más importantes son las siguientes:

— Gestionar y planificar la memoria del sistema, de manera que realiza una división de la misma, asignando a cada tarea un "banco de memoria".

— Distribuir los recursos entre todos los usuarios. Asigna el tiempo de utilización del procesador a cada uno de los trabajos a realizar "simultáneamente", de modo que resulte transparente a cada usuario, y éstos tengan la "ilusión" de disponer de un sistema independiente para cada uno.

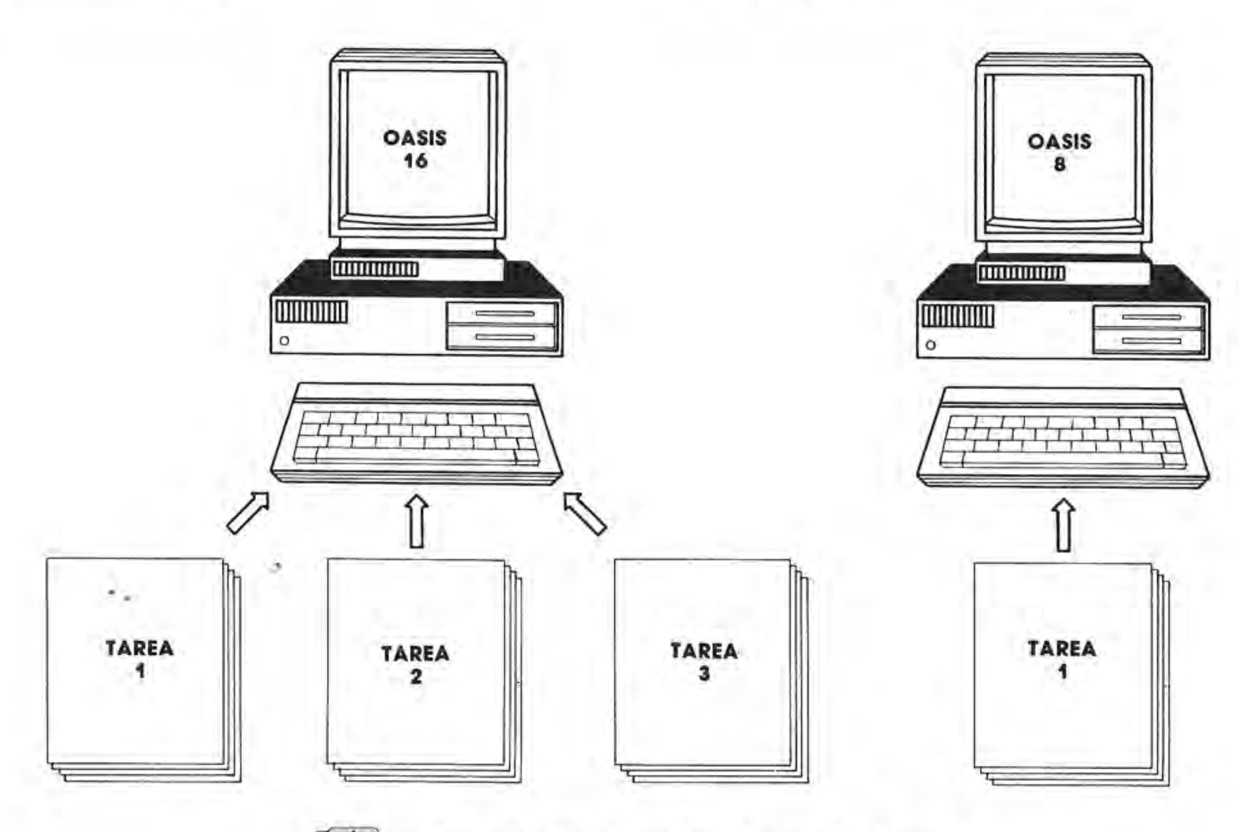
— Gestionar la entrada y salida de datos.
— Inicializar los dispositivos periféricos.
— Detección y tratamiento de errores.

2. *El CSI (Intérprete de Cadenas de Comandos).* Su misión es realizar el control de acceso al sistema, control de acceso a los programas de usuario y ejecución de procedimientos de entrada y salida. Cuando se introduce un comando en el sistema, el CSI busca en el disco el programa cuyo nombre coincida con el comando introducido, transfiriéndole el control para su ejecución. En caso de no encontrar ningún comando o programa cuyo nombre coincida con el introducido por el usuario, se visualiza en pantalla un mensaje de error.

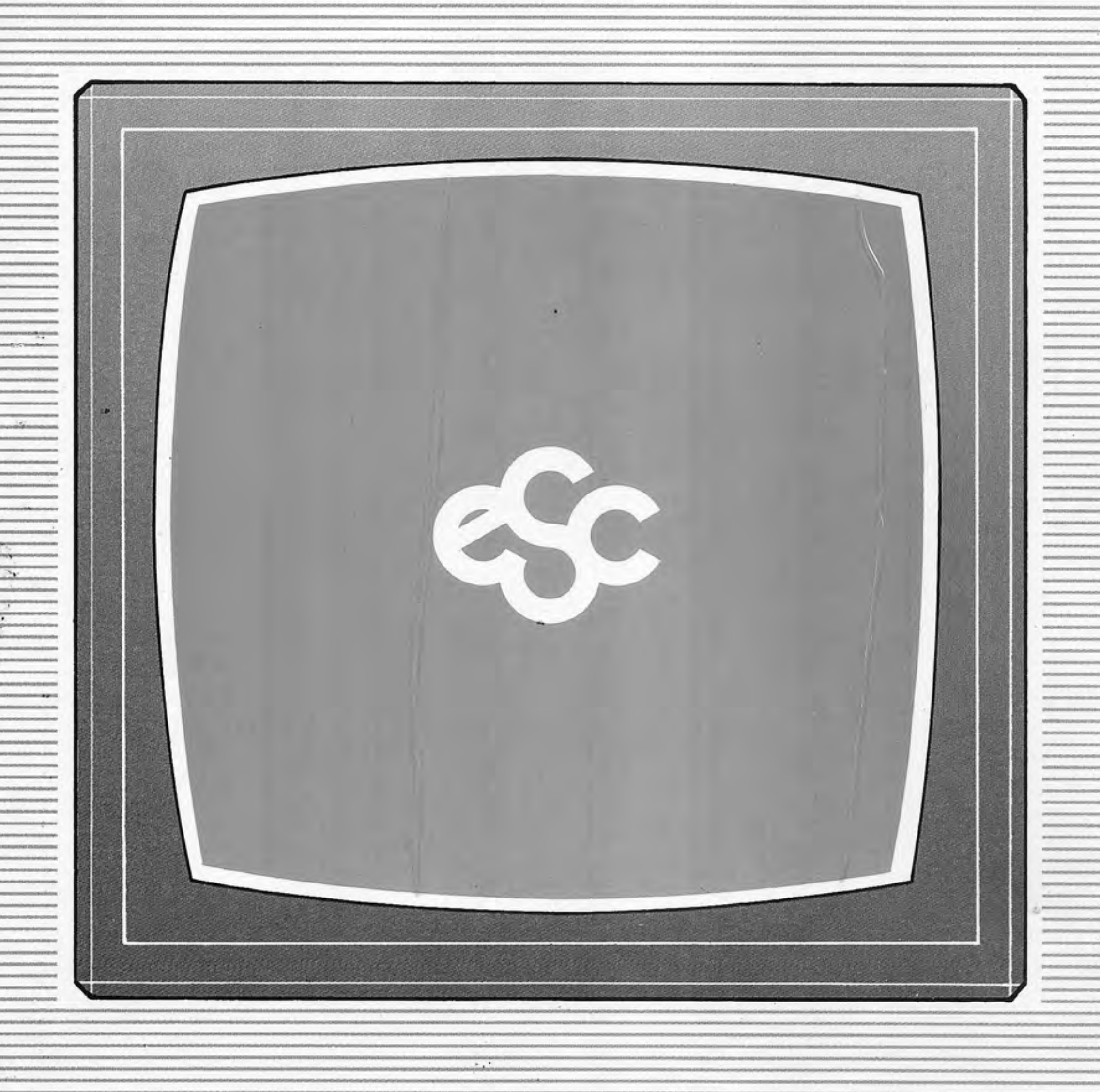
El CSI es similar al CCP de CP/M o al COMMAND.COM de MS/DOS.

3. *Utilidades para el control de procesos y procesadores de lenguajes.* Entre estos últimos se encuentra una versión de BASIC (en versión intérprete y en compilador), especialmente preparada para el desarrollo de programas de gestión. Asimismo, se dispone de compiladores para los lenguajes Pascal y COBOL.

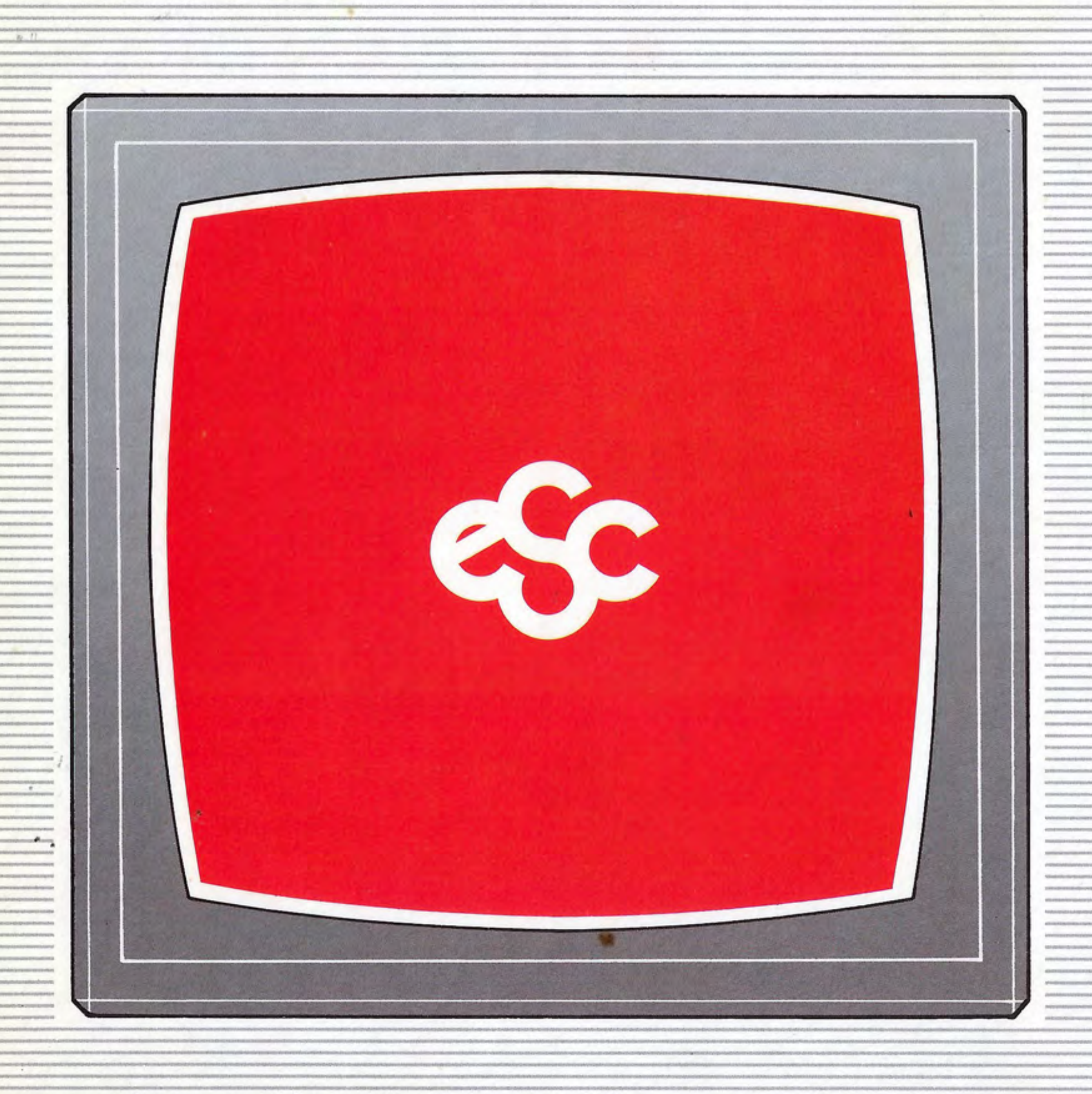
Para el control de procesos, OASIS 8 dispone de un lenguaje especial llamado EXEC. En esencia, permite hacer "programas de comandos", de manera que se ejecutan como si fueran introducidos uno a uno por teclado. Tiene, además, la posibilidad de incluir ejecución condicional y ejecución iterativa.



Una de las diferencias más grandes entre OASIS-8 y OASIS-16 es la capacidad multitarea de este último.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼